Combat Identification of Synthetic Aperture Radar Images using Contextual Features and Bayesian Belief Networks

THESIS

John X. Situ, Captain, USAF

AFIT-OR-MS-ENS-12-24

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-OR-MS-ENS-12-24

COMBAT IDENTIFICATION OF SYNTHETIC APERTURE RADAR IMAGES USING CONTEXTUAL FEATURES AND BAYESIAN BELIEF NETWORKS

THESIS

Presented to the Faculty

Department of Operational Sciences

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Operations Research

John X. Situ, BS

Captain, USAF

March 2012

AFIT-OR-MS-ENS-12-24

COMBAT IDENTIFICATION OF SYNTHETIC APERTURE RADAR IMAGESUSING CONTEXTUAL FEATURES AND BAYESIAN BELIEF NETWORKS

John X. Situ, BS
Captain, USAF

Approved:

_______//SIGNED//___________________	_15 March 2012__
Mark A. Friend, PhD (Chairman)	Date
_______//SIGNED//____________________	_15 March 2012__
Kenneth W. Bauer, PhD (Member)	Date

AFIT-OR-MS-ENS-12-24

Abstract

Given the nearly infinite combination of modifications and configurations for weapon systems, no two targets are ever exactly the same. SAR imagery and associated High Range Resolution (HRR) profiles of even the same target will both have different signatures when viewed from different angles. To overcome this challenge, data from a wide range of aspect and depression angles must be used to train pattern recognition algorithms. Alternatively, features invariant to aspect and depression angles must be found. This research uses simple segmentation algorithms and multivariate analysis methods to extract contextual features from SAR imagery. These features used in conjunction with HRR features improve classification accuracy at similar or extended operating conditions. Classification accuracy improvements achieved through Bayesian Belief Networks and the direct use of the contextual features in a template matching algorithm are demonstrated using a General Dynamics Data Collection System SAR data set.

AFIT-OR-MS-ENS-12-24

To my wife, who patiently supported me through this endeavor.

Thank You!

Acknowledgements

I would like to express sincerest gratitude to my thesis advisor, Dr. Mark Friend and my committee member, Dr. Kenneth Bauer for their guidance and encouragements during my research. I like to specially thank Dr. Friend for his mentorship and patience. In addition, I would like to thank Mr. Trevor Bihl for his efforts and aid. Lastly, thank you to all my classmates for all their support.

John X. Situ

Table of Contents

List of Figures

Figure Page

Figure Page

List of Tables

COMBAT IDENTIFICATION OF SYNTHETIC APERTURE RADAR IMAGES USING CONTEXTUAL FEATURES AND BAYESIAN BELIEF NETWORKS

I. Introduction

Since the dawn of warfare, military objectives have been simple. They revolved around the actions of locating, identifying and engaging the enemy. Until the late 20th century, most combat occurred within visual range of the opponent, where Combat Identification (CID) of a target can almost be of a certainty before engaging. Due to the rapid technological advancements in weaponry, the engagement ranges on most of our advanced weapons have evolved past the range human abilities can identify. Advanced radio detection and ranging (RADAR) technologies and satellite imagery provide the ability to locate and identify targets from a great distance. The issue became how to process the massive amount of information that must be processed in order to find and identify targets within the collected imagery. It takes time and manpower for a human analyst to process such information. On a battlefield filled with hundreds or thousands of targets, it would be an overwhelming and time consuming task for humans. One solution to this problem was to teach machines how to locate objects of interest and identify them with little or no human interactions. This process is known as Automatic Target Recognition (ATR).

In order for humans to recognize and classify an object, we must first have samples or descriptions of what each of the possible classes of objects look like. We capture this data from our lifetime of experiences and store them in our memories. The same is true for machines; they need a library of distinguishing characteristics (such as length, height, color, etc.) of object classes, also known as features. However, an object has more than one set of features for identification, and their features can change according to angle of perception of the sensor. For example, a car looks different from the sides than from the front, the dimensional measurements and general shapes are different. These features can also differ by through deliberate modifications. That same car will also look different with its trunk open or with radio antennas installed. The human mind is exceptionally efficient at intuitively interpolating those feature changes, machines can currently only interpolate from what it is provided. In order to achieve high levels of classification accuracy, we need to provide machines with data on every possible angle of view for all object classes. Given the growing number of weapon systems around world, it would be near impossible capture such information for every weapon system to in order achieve high levels of classification accuracy required for CID.

The Department of Defense (DoD) defines CID as "The process of attaining an accurate characterization of entities in a combatant's area of responsibility to the extent that high-confidence, real-time application of tactical options and weapon resources can occur. The objective of CID is to maximize combat/mission effectiveness while reducing total casualties (due to enemy action and fratricide)" [6]. The 1999 Kosovo air campaign is an example of the high cost to a modern conflict due to inaccurate combat identification of ground targets. Of the 120 identified tanks NATO claims to have destroyed, only 14 were verified as tanks post conflict[7]. It is estimated that for only one Serbian ground target was destroyed for every 72 sorties flown by NATO[9]. Unreliable CID can not only create unnecessary causalities, but also needlessly expend valuable resources. An official USAF study of the Kosovo air campaign suggested that "The Air Force must continue to investigate new technologies and techniques for locating hidden or dispersed ground force elements with targeting quality accuracy, and rapidly passing that data to the 'shooters'"[11].

One solution is to find features that remain relatively constant for different angles or modifications. This research uses contextual features in addition to previously proven High Range Resolution (HRR) features of Synthetic Aperture Radar (SAR) images to improve classification accuracy. In addition, a Bayesian Belief Network is employed to fuse the outputs from multiple classification algorithms and contextual information to produce an overall combined classification result with higher levels of accuracy.

Chapter 2 will provide an overview of hyperspectral imagery, synthetic aperture radar and pattern recognition algorithms related to this research and summaries the previous works in these areas. Chapter 3 contains the research methodology, proposed algorithms, and measures of evaluation used in this research. Chapter 4 contains the results and analysis. Chapter 5 summarizes this work and provide a conclusion to the research conducted, discusses the contributions provided and potential future research.

II. Literature Review

This chapter provides a summary of the previous contributions related to the fields of CID and ATR. Starting with a brief overview of Hyperspectral Imaging (HSI) and SAR basics, the sections also include introductions to pattern recognition approaches, template classification algorithms, fusion methods, examples of Bayesian Belief Networks, and the application of contextual information.

2.1 Hyperspectral Imaging (HSI) Basics

Digital images capture information from the electromagnetic (EM) spectrum and display their relative intensities. The whole EM spectrum includes for instance, UV visible, X-rays, gamma rays, microwaves, and radio waves. The most familiar images are photos in the visible EM region containing the visible three bands human eyes can see: red, green, and blue. Each of these three bands contains a captured image with relative intensity of the corresponding EM radiation. The color image is a result of all three images overlaid into one. A hyperspectral image contains information from other portions of the EM spectrum, using up to 250 bands. These bands span the visible (.4 μm to .7 μm), near-infrared, and mid-infrared (.7 μm to 2.5 μm)[20]. Since materials reflect electromagnetic energy differently based on their composition, they have distinctive reflection patterns at specific wavelengths. HSI utilizes this property to identify and distinguish between different materials.

The data contained in a hyperspectral image can be represented as a p-dimensional vector for each of the pixel in the image, where p is the number of spectral bands used. For processing and analysis, the data is stored as a hyperspectral data cube; a three-dimensional array where m and n represent the spatial location and p is the spectral dimension.[28]

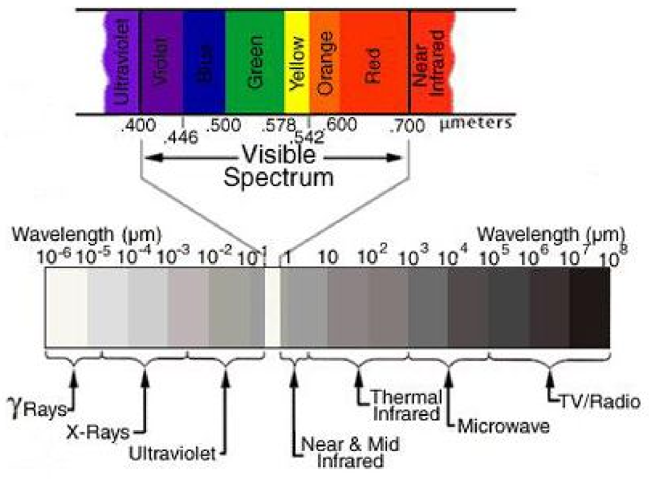


Figure 1. ***Electromagnetic Spectrum*, figure reprinted from[10]**

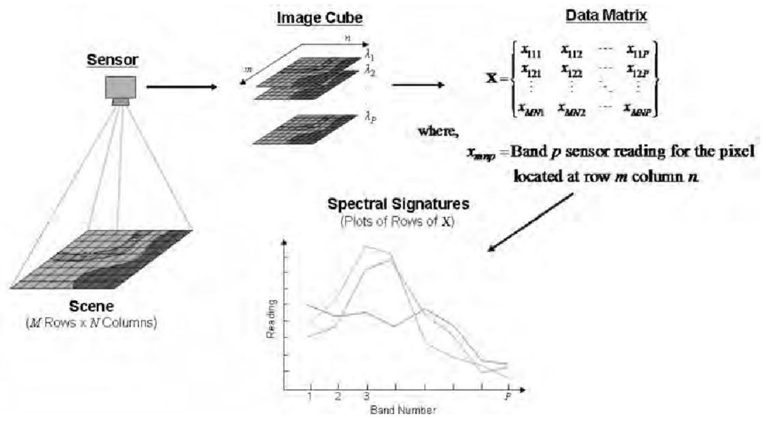


Figure 2. ***HSI process data and representation*, figure reprinted from[28]**

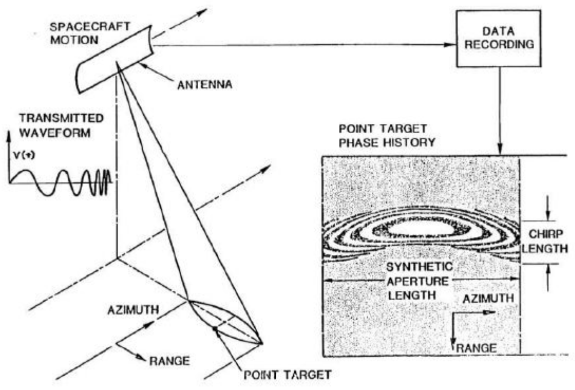


Figure 3. ***Synthetic Aperture Radar*****, figure reprinted from[29]**

2.2 Synthetic Aperture Radar (SAR)

HSI makes use of the radiation naturally emitted or reflected by any surface and is considered a passive image sensor system. The principal limitation of passive sensors is the lack of an independent source of radiation, its imaging capabilities are reduced significantly when clouds or fog cover the area of interest. Radar is an active image sensor requiring its own transmitting system, which receives the backscattered radiation from the illuminated surfaces. Active systems are not dependent on sunlight and allow for day, night and all-weather imaging. One limitation of to Real Aperture Radars (RAR) is low resolution. At long ranges, radar antennas would need to be several hundred meters wide to achieve resolution on the order of magnitude of meters. To overcome this limitation, a small antenna can be moved along a path, process the received signals, and synthesize a very large antenna[12].

SAR is an airborne or space-borne side-looking radar system that utilizes the flight path of the platform to simulate an extremely large antenna or aperture electronically, and generate high-resolution remote sensing imagery. The radar illuminates an area by transmitting pulses of microwave energy. The pulses are reflected back by objects and then collected by the radar receiver. The distance or range from the platform to the illuminated objects can be calculated by measuring the time difference between energy transmission and reception of the reflected energy[12]. The range resolution is a function of the effective pulse-width τ multiplied by the speed of light c and divided by two [27]

$$\text{Range Resolution} = \frac{c\tau}{2}. \tag{1}$$

The direction orthogonal to the radar beam is known as azimuth, it is also the linear trajectory the sensor is moving along, or the flight path. Two targets at a given range can be resolved, as long as they are not in the radar beam at the same time. The azimuth resolution of a radar system is a direct function of the radar wavelength λ, the target range R and an inverse function of antenna dimension D [27]

$$\text{RAR Azimuth Resolution} = \frac{\lambda}{D} \cdot R. \tag{2}$$

For a RAR system at any particular range, the only ways to lower the azimuth resolution are to either increase the frequency$(1/\lambda)$, or increase the dimension D of the radar[27]. Both are impractical solutions for space borne or air borne systems due to the weight and cost required to implement them.

By moving the radar along a path, a Doppler history of the target energy returns is generated and resolution can be determined by using that record. The distance the moving radar travels while an object is in view becomes the dimension of the radar, or the synthetic aperture. Since the synthetic aperture is directly proportional to the range, the result is a SAR system that can produce an image whose azimuth resolution is independent of wavelength and target range[27]. The SAR azimuth resolution is function of only D, the travel distance the radar

$$\text{SAR Azimuth Resolution} = \frac{D}{2}. \tag{3}$$

2.2.1 HRR Profiles.

SAR systems combine signal returns over different times and post-process them using Fourier transformations for frequency sampling, creating one image of the target. The difficulty with SAR imaging increases with the addition of moving targets. As a target moves, its relative position from the radar changes and the resulting images are smeared and displaced[33]. One approach is to process the SAR image chips into one dimensional High Range Resolution (HRR) profiles. Using HRR profiles for recognizing moving targets significantly enhances target to noise ratio, via Doppler filtering and clutter cancellation[32]. Due to the unavailability of a sufficiently large set of processed images of ground moving targets, this research uses stationary SAR data to emulate the data provided by a rand function in HRR radar mode against moving targets. Figure 4 describes the process used to create HRR profiles from SAR images for this research and previous research. The algorithms used originated from the Air Force Research Laboratory(AFRL) at Wright Patterson AFB[32].

2.2.2 SAR data.

For most of the previous studies using HRR profiles, target features came from Moving and Stationary Target Acquisition and Recognition (MSTAR) data[1, 13, 19, 31] and General Dynamics Data Collection System (DCS) SAR data[13]. The DCS data set used in this research originated from AFRL and was collected from a twin engine Convair 580 aircraft during May 2004 at Eglin AFB using a General Dynamics DCS X-band SAR radar operating in spotlight mode. The DCS radar imagery was

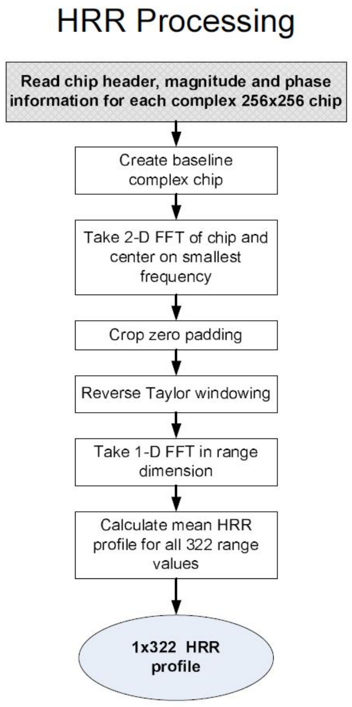

HRR Processing
Read chip header, magnitude and phase information for each complex 256x256 chip
Create baseline complex chip
Take 2-D FFT of chip and center on smallest frequency
Crop zero padding
Reverse Taylor windowing
Take 1-D FFT in range dimension
Calculate mean HRR profile for all 322 range values
1x322 HRR profile

Figure 4. ***Conversion of SAR images into HHR profiles process*, figure reprinted from[13]**

collected at a resolution of 1.0 ft in both HH-polarization and VV-polarization. Data was collected on 15 different targets, detailed in Table 1. All targets were stationary in open areas with no concealment. The resulting SAR chips used for this work are images of 256 x 256 pixels. The DCS data consists of 724 SAR image chips collected each for 15 target. Within each SAR chip, there is an image of the target in HH polarization and one in VV polarization, resulting in a data set is a 15 by 2 by 724 images. The data was collected at a desired depression angle of between 6 and 8 degrees. Each SAR image chip is first processed into an HRR profile as described in section 2.2.1. Next, the HRR profiles are then ordered by aspect angle and interpolated over 360 degrees into one degree increments; For each of the HRR profiles 10 features are generated to characterize the target. Each target's data is then separated into 15 degree non-overlapping wedges. These 15 degree wedges are used to create prototype templates for each target resulting in 24 non-overlapping templates,

Group	Type	Target Description	Track	Wheels	Gun	Label
Hostile	SCUD	Single Large Missile	N	8	N	TOD
	SMERCH	MLRS Scud Confuser	N	8	N	OH1
	SA-6 Radar	Similar to SA-6 TEL	Y	0	N	OH2
	T-72	Main Battle Tank	Y	0	Y	OH3
	SA-6 TEL	3 Medium SAMs	Y	0	N	OH4
Friendly and Neutral	Zil-131	Medium Budget Truck	N	4	N	FN1
	HMMWV	Jeep like SUV	N	4	N	FN2
	M113	Armored Personnel Carrier	Y	0	Y	FN3
	Zil-131	Small Budget Truck	N	4	N	FN4
	M35	Large Budget Truck	N	4	N	FN5
Out of Library	SA-8 TZM	SA-8 Reload vehicle	N	6	N	OOL1
	BMP-1	Tank w/small turret	Y	0	Y	OOL2
	BTR-70	8-wheeled transport	N	8	N	OOL3
	SA-13	Turret SAMs	Y	0	N	OOL4
	SA-8 TEL	Integrated radar exposed SAMS	N	6	N	OOL5

Table 1. Table of the 15 targets in DCS data along with associated parameters

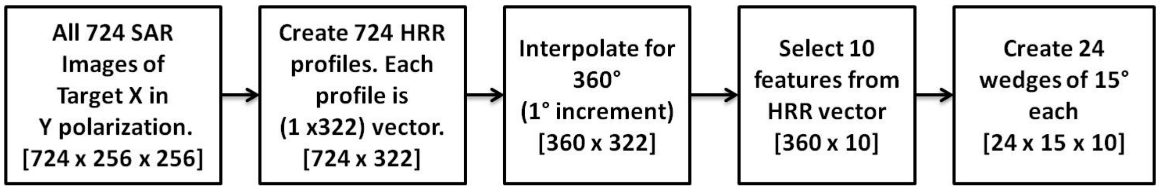

All 724 SAR Images of Target X in Y polarization. [724 x 256 x 256]
Create 724 HRR profiles. Each profile is (1 x322) vector. [724 x 322]
Interpolate for 360° (1° increment) [360 x 322]
Select 10 features from HRR vector [360 x 10]
Create 24 wedges of 15° each [24 x 15 x 10]

Figure 5. ***Template development process***

spanning the 360 observable aspect angles for each target. Figure 5 illustrates the data organization process.

2.3 Pattern Recognition

Pattern recognition or classification is systemically selecting targeted data via important features from a background of noisy data. There are two types of pattern recognition; the first is supervised classification, in which the target can be identified as a predefined class of object. The second is unsupervised classification, in which the target is assigned to an unknown class. Jain et al.[17] defined the four best known approaches to pattern recognition: The first is template matching, one of the simplest and earliest approaches. Using representative data from each class, a library of prototype templates are created during classifier training. and the object of interest, also known as an exemplar, is compared to the templates. Once the algorithm has been trained, exemplars or objects of interest, are compared to the template via a similarity metric, such as a Mahalanobis distance. The object being classified, with its data organized as a multivariate vector, $x = (x_1, x_2, ..., x_N)^T$ is evaluated against a prototype template with a mean of $\mu = (\mu_1, \mu_2, ..., \mu_N)^T$, S is the covariance matrix. A Mahalanobis distance is the following:

$$D_{\mathrm{M}}(x) = \sqrt{(x-\mu)^T S^{-1}(x-\mu)}. \tag{4}$$

The class template with the shortest Mahalanobis distance to the exemplar of interest is identified as the objects class.[17] The second approach is statistical, in which each pattern is represented in terms of a number of features or measurements, which can then be viewed as a point in a d-dimensional vector space. The goal in the statistical method is to choose features such that the pattern vectors occupy disjoint spaces in the d-dimensional vector space. A linear discriminant analysis is an example of a statistical method. The third approach is the syntactic approach, which takes the perspective that a pattern is a complex hierarchical system with simple sub-patterns, known as primitives. The complex pattern is represented in terms of the interrelationships among the primitives. In Jain's[17] analogy for syntactic approach, patterns are the sentences of a language and primitives are the alphabet of the language. An entire language of sentence, a large set of patterns, can be described by a small number of primitives and grammatical rules. The grammatical rules for a language can be obtained from a set of training data. A person can understand a language to certain accuracy given a training set of sentences with its meaning, they can identify

Approach	Representation	Recognition function	Typical criterion
Template matching	Samples, pixels, curves	Correlation, distance measure	Classification error
Statistical	Features	Discriminant function	Classification error
Syntactic or structural	Primitives	Rules, grammar	Acceptance error
Neural networks	Samples, pixels, features	Network function	Mean squre error

Table 2. *Approaches to Pattern Recognition*, table adapted from[17]

the alphabets or grapheme and infer a grammatical structure. When given a new sentence, a person will identify the alphabets and their interrelationship, and then they will figure out the meaning of the sentence from their library of grammatical rules. The fourth and last approach listed by Jain are Neural Networks, which can be viewed as massive parallel computing systems consisting of an extremely large number of simple processors with many interconnections. This research uses Probabilistic Neural networks (PNN) for classification, further explained in Section 2.5. Table 2 is a summary of the four approaches.

2.4 Template Classification

This research follows[1, 19, 13, 31], focusing on making improvements to classification of SAR images using template based classification. The SAR images of a target of interest captured by a data collection system vary significantly depending on the geometry of the collection and are a function of the angles of view from the data collection system, known as aspect and depression angles. An aspect angle describes the direction which faces the radar sensor. Figure 6 illustrates some example aspect angles. Using a reference view looking directly over a target and characterizing the

360 degrees around it, 0 ° indicates the data is collected with the sensor in front of the target and at 90 ° data is collected on the right side of the target. The Depression angle is the angle between the horizontal plane and the path from the radar to the target, as illustrated by figure 7.

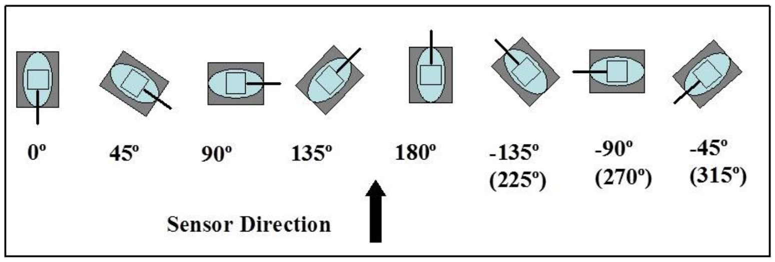


Figure 6. ***Aspect Angle according to DCS data*****, figure reprinted from**[19]

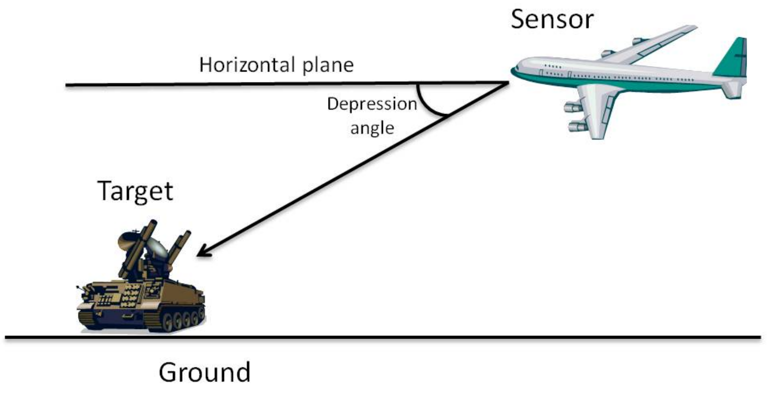


Figure 7. ***Illustration of Depression Angle***

SAR images of the same target collected at different aspect and depression angle from the sensor are usually significantly different. The angle of view impacts how an object reflects radar energy and creates different signal returns, resulting in different images and different HRR profiles. The unique characteristic of SAR imagery poses a challenge for an ATR system using SAR. One solution is to create multiple templates of the same target prototype at different aspect angles.

In this research, assumptions are made concerning the accuracy of aspect angle estimates. These assumptions are reflected in the size and number of wedges used for template matching. When comparing an object of interest against a template, the perceived aspect angle is assumed to be within $\pm 22.5^\circ$ of the true aspect angle; this covers three wedges of 15° each, so the object is matched against three templates, one at the assumed aspect angle and each of it's adjacent wedges. As described in section 2.2.2, each library target class will have 24 templates of 15 degrees each. Figure 8 illustrates the aspect angle assumption wedges of 15° each. In the figure, the red wedge is the template that contains the assumed aspect angle of target, the adjacent yellow wedges are the other two wedge templates being compared.

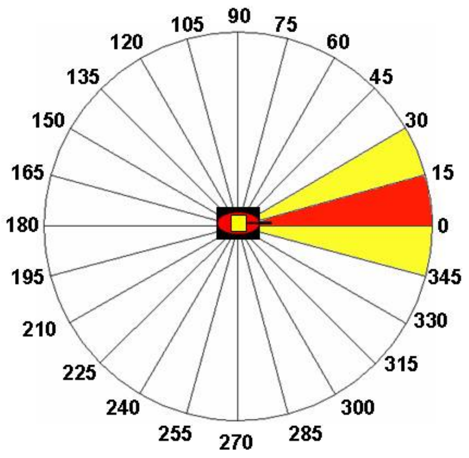


Figure 8. ***Aspect angle assumption*** 15° ***wedges. The red wedge contains assumed aspect angle*****, figure reprinted from[13]**

The squared Mahalanobis distance is measured between the exemplar's 10 features and each of the three templates 10 features. For a classification system with N target classes, the minimum of these squared Mahalanobis distances is recorded in a N-dimensional vector, along with the corresponding template number on the second N-dimensional vector. For example, in classification with 4 target classes, each class has three 3 Mahalanobis distances from 3 wedges. Class 1 has values of $[31.5, 13.4, 41.5]$, Class 2 has $[11.6, 14.7, 3.8]$, Class 3 has $[64.3, 24.1, 12.7]$, Class 4 has $[7.7, 16.5, 32.9]$; the resulting 4-dimensional vector from the minimum distance of each class is [13.4, 3.8, 12.7, 7.7]. The winning class in the case is class 2, which has the minimum distance of 3.8, the exemplar has the shortest Mahanlanbois from class 2 wedge 3. Figure 9 graphical illustrates the example.

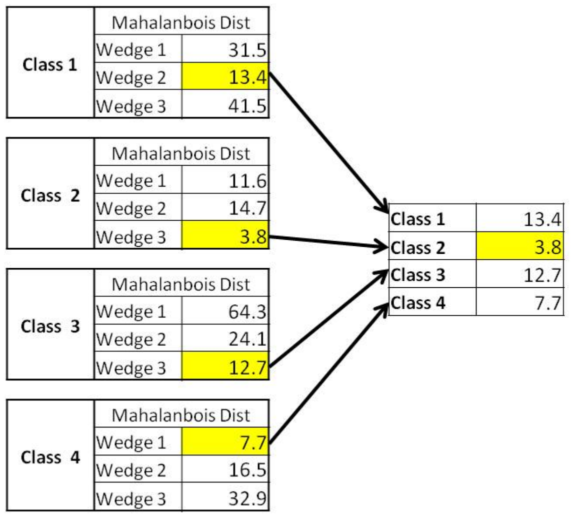


Figure 9. ***Example of multiple wedge template matching using Mahalanbois distance***

Template matching uses these two vectors to classify the object of interest. In a traditional forced decision template matching classifier, the exemplar's class is determined using the two vectors. However, previous AFIT research[1, 19, 13, 31] generated new options; instead of forcing a decision, an out-of-library (OOL) or non-declaration (NDEC) decision can be made.

2.4.1 Out of Library Targets.

An an Out of Library(OOL) decision is made when an exemplar differs significantly from all of the library templates. An object of interest is marked as OOL if the winning class scores exceeds some previously establish threshold, indicating that it may not have a template in classifiers library. This usually happens when a new weapon system is fielded and no data has been collected on the object of interest. It could also be the classifier trying to identify an object outside the bounds of its design. For example, a system designed to identify ground vehicles does not have aircraft templates in its library.

Albrecht's[1] OOL algorithm creates an exemplar's OOL posterior probability, PP_{OOL}, as a function of the N-dimensional class posterior probability vector. For an exemplar, the posterior probability vector X_{post} are ordered from largest to smallest. The element with the highest element is placed in the first spot, descending to the lowest element placed in the last spot. For example, a classifier with 10 target classes produces the following 1x10 posterior probability vector [31]:

$$X_{post} = [0.9, 0.01, 0, 0.05, 0, 0.02, 0, 0, 0.02, 0] \tag{5}$$

ordering from greatest to smallest produces:

$$X_{ord} = [0.9, 0.05, 0.02, 0.02, 0.01, 0, 0, 0, 0, 0] \; . \tag{6}$$

Two thresholds for OOL determination are calculated based on predetermined parameters

$$X_{OOL} = \sum_{i=2}^{\theta_{OOL}^{(1)}} X_{ord}(i) \;\; , \tag{7}$$

where $\theta_{\mathrm{OOL}}^{(1)}$ is n^{th} ordered number of values from the sorted posterior vector. For example if $\theta_{\mathrm{OOL}}^{(1)} = 5$, $X_{ord}(\theta_{\mathrm{OOL}}^{(1)})$ would be 0.01 and X_{OOL} would be (0.05+0.02+0.02+0.01). X_{OOL} would then be evaluated against $\theta_{\mathrm{OOL}}^{(2)}$ to determine PP_{OOL}

$$PP_{OOL} = \begin{cases} 0 & \text{if } X_{OOL} < \theta_{OOL}^{(2)} \\ f(X_{OOL} - \theta_{OOL}^{(2)}) & \text{if } X_{OOL} \geq \theta_{OOL}^{(2)} \;\; , \end{cases} \tag{8}$$

where $\theta_{\mathrm{OOL}}^{2}$ is threshold from sub-optimization routine and

$$d = X_{\mathrm{OOL}} - \theta_{OOL}^{2} \tag{9}$$

$$f(d) = \frac{2}{1 + e^{-10d}} \quad . \tag{10}$$

After PP_{OOL} is determined, it is concatenated to the end of the N-element estimated posterior vector, X_{post}, as the N+1 element and normalized to produce the OOL estimated. For the previous example, N = 10, so PP_{OOL} would be the 11^{th} element in the new concatenated vector.

Friend's [13] OOL median/MAD method uses a squared Mahalanobis distance from the set of in-library training data correctly identified as the OOL threshold. For each target class, a training data set of in-library exemplars are put through the classifier. All correctly identified training exemplars are extracted along with their corresponding squared Mahalanobis distances. The median and mean absolute deviation (MAD) distances are used to calculate the class OOL threshold. Friend hypothesized that since the squared Mahalanobis distances for the in-library training set will be less than the distances for test exemplar, a constant alpha multiplier needs to be included. The determination for OOL is the following[13]:

$$OOL_{record} = \begin{cases} 1 & \text{if } Mdist_{template,record} > median_{template} + \alpha \cdot MAD \\ 0 & \text{if } X_{OOL} \geq \theta^{(2)}_{OOL} \quad . \end{cases} \tag{11}$$

If the winning Squared Mahalanobis distances for a test exemplar is greater than the winning templates class OOL threshold, than the exemplar is declared OOL. Work on this method was never completed, due the fact that an effective MAD multiple alpha needs to be found through experimentation with test data that includes OOL records, which is an unacceptable data requirement.

Friend's[13] OOL Quantile Method sets the OOL threshold as n^{th} quantile highest squared Mahalanobis distance from the set of in-library training data correctly identified. Similar to median/MAD method, for each target class, a training data set of in-library is put through the classifier. All correctly identified training exemplars are extracted along with their corresponding squared Mahalanobis distances. The OOL threshold, θ_q is the nth quantile sorted squared Mahalanobis distance. In most cases, Friend[13] states experimentation points toward using the quantile value of 1, which equals to the 100th percentile or the maximum score. If a test exemplar's winning squared Mahalanobis distance is higher than the winning templates θ_q, then the exemplar is declared OOL.

Turbaugh's[31] heuristic OOL methodology follows the same basic steps as Friend's OOL median/MAD methods. Using a training set of in-library targets, the similarity metrics L, were computed for each correctly identified training exemplars. The threshold values x_L were then computed, in a similar fashion like the other heuristic methods. A threshold for each target class x_L, was then generated using the mean, μ and standard deviation, σ over all values of L for the correctly identified exemplars in each target class training set. A test exemplar will be assigned an OOL label if the L between the winning template and the test record exceeds the threshold x_L corresponding to that target class, where $x_L = \mu + \sigma$. Like Friend[13], rather than using a single standard deviation as the threshold distance, a multiple α can be implemented to σ so that the threshold value is:

$$x_L = \mu + \alpha \cdot \sigma \quad . \tag{12}$$

If an exemplar's winning metric is higher than the winning templates x_L, than the exemplar is declared OOL, as displayed in figure 10, a diagram of Turnbaugh's OOL method.

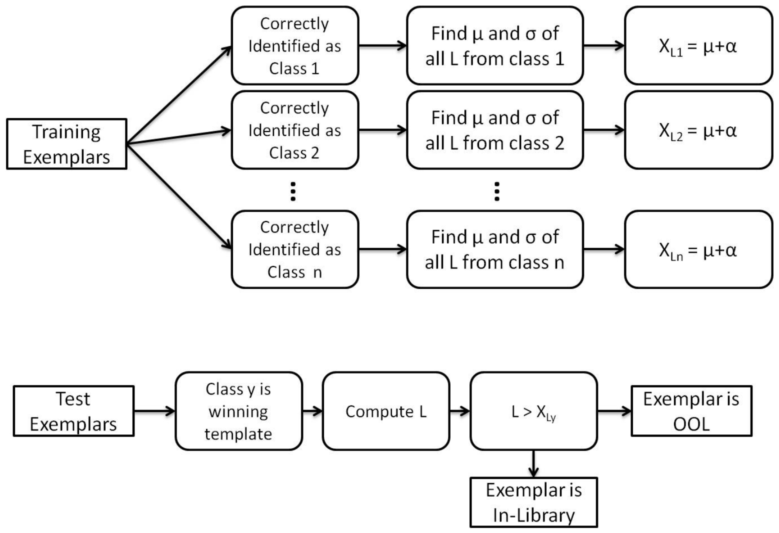

Training Exemplars
Correctly Identified as Class 1
Correctly Identified as Class 2
Correctly Identified as Class n
Find μ and σ of all L from class 1
Find μ and σ of all L from class 2
Find μ and σ of all L from class n
$X_{L1} = \mu+\alpha$
$X_{L2} = \mu+\alpha$
$X_{Ln} = \mu+\alpha$
Test Exemplars
Class y is winning template
Compute L
$L > X_{Ly}$
Exemplar is OOL
Exemplar is In-Library

Figure 10. ***Turnbaugh's OOL Method***

2.4.2 Non-Delcaration Targets.

If an exemplar is similar to multiple target classes, it can lead to a winning class score being relatively close to some of the other classes. The closer the scores, the greater chance of misclassification and less confidence there is in declaring a class. However, thresholds can be established to make a Non-Declaration(NDEC) to reduce misclassification or critical errors. A decision of non-declaration can made for an exemplar. Leap [21] notes as the threshold increases, the number of decisions made decreases, withholding less confidence decisions and resulting in an increase in overall engineering confidence. As exemplars are removed from classification consideration and declared NDEC, the classification accuracy tends to increase. This causes engineering confidence to increase. The number of exemplars or the sample size, used to train the classifier remains constant across the confidence threshold space. However, the percentage of declarations continues to decrease as the confidence threshold increases. Thus, there is a tradeoff between more confident decisions and number of decisions made. In this section, previous work in NDEC are be explored.

Chow[3], stated that the performance of pattern recognition system is characterized by its error and rejection tradeoff. By withholding exemplars which are difficult to classify, classification accuracy can be improved. Chow's rejection rule evaluates the posterior probability of the winning class against a single threshold for all classes. Using a classifier with N classes, the exemplars winning class i is computed as the class with the maximum posterior probability given x. All exemplars with $P(w_i|x)$ less than the threshold T is rejected under this rule:

$$x \notin w_i \quad \text{if} \quad \max_{k=1,2,\dots N} P(w_k|x) = P(w_i|x) < T \quad \text{where } T \in [0,1] \ . \tag{13}$$

Fumera et al.[14] enhanced Chows work by allowing a different threshold for each class, instead of the all classes using the same threshold. The exemplar x is rejected if the winning posterior probability for class i $P(w_i|x)$ is less than the threshold for i

$$\max_{k=1,2,\dots N} \hat{P}(w_k|x) = \hat{P}(w_i|x) < \theta_i \quad \text{where } \theta_i \in [0,1] \ . \tag{14}$$

The exemplar is given the label w_i if the winning posterior probability for class i $\hat{P}(w_i|x)$ is greater or equal the threshold for i

$$\max_{k=1,2,\dots N} \hat{P}(w_k|x) = \hat{P}(w_i|x) \geq \theta_i \ . \tag{15}$$

Laine[19] and Albrecht[1] utilized a NDEC method based on receiver operating characteristic (ROC) curve analysis to determining a threshold. A ROC curve is a graphical tool to conduct sensitively studies of trade off between true-positive rates and false positive rates as the threshold θ is varied from 0 to 1.[31] Their NDEC/rejection region is defined by the interval $(\theta_{ROC}, \theta_{ROC}+\theta_{REJ})$, where $\theta_{REJ} > 0$. An example of the ROC and rejection thresholds is included in figure 11. It shows

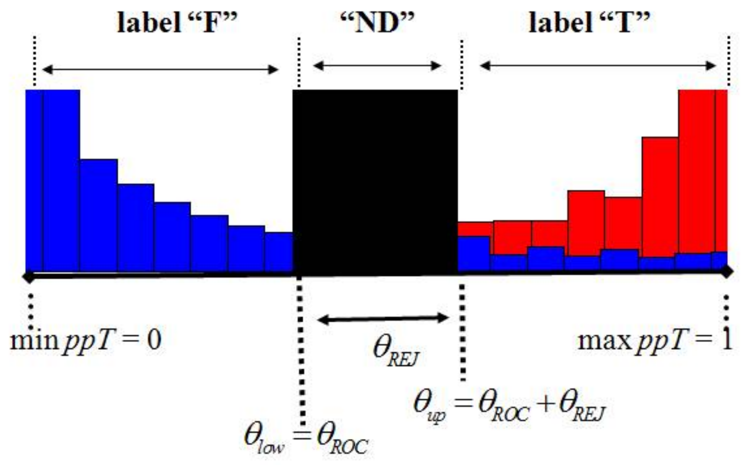


Figure 11. ***Rejection region created by the paramters θ_{ROC} and θ_{REJ}*, figure reprinted from[31]**

the declaration labels for a set of two-class data represented by the red and blue histograms of along an x-axis corresponding to the posterior probability.

Friend's [13] entropy method uses information theory to evaluate whether a classifier has enough information to make a decision. In his methods. entropy is calculated from the distribution of correctly indentified target exemplars in training. For a classifier with N target classes, the classification of each target class is a random variable, Y, and its 1 x N vector of posterior probability is a probability mass function, p_Y, where $p_Y(y) = Pr\{Y = y\}$, $Y \in 1...N$. Using the set of training exemplars, each of the in-library target classes entropy H can be computed as:

$$H(Y) = \sum_{i=1}^{N} p_y(y_i) log_2 \left(\frac{1}{p_y(y_i)} \right) = \sum_{i=1}^{N} -p_y(y_i) log_2(p_Y(y_i)) = -E[log_2(p(Y))] \quad (16)$$

After the training phase, the NDEC threshold is established by selecting a user defined quantile of class' entropy distriubtion. During testing, any test exemplar with an entropy higher than its winning class pre-established entropy NDEC thresholdl receives a NDEC label.

Friend's [13] KullbackLeibler (KL) method is based on KullbackLeibler distance or relative entropy. KL distance is a measure of the difference between two probability distributions. In a classification system with N number of classes, the target class is as a represented random variable, X and p_X is the probability mass function, where $p_X(x) = Pr\{X = x\}$, $X \in R^n$ is a 1 x N vector of the posterior probability estimation of an exemplar belonging to each of the N classes. The truth, q_X is the true distribution function, with an entropy value of 0. Friend defines the KL distance between p_X and q_X as

$$D(p_x, q_x) = \sum_{i=1}^{N} p_X(x_i) log\left(\frac{p_X(x_i)}{q_X(x_i)}\right) = -log(q_x(x_i*)) \tag{17}$$

where the winning class template is i^*[13]. The algorithm for Friend's KL distance method is similar to his entropy NDEC method with the same basic steps. When computing the NDEC thresholds for each target class template, simply substitute KL distance for Entropy.

Turbaugh's [31] NDEC method extends the works of Fumera et al.[14] and Friend[13], using a single non-declaration threshold for each class. In his non-declaration method, a exemplar x is labeled NDEC if

$$\max_{k=1,2,...N} \hat{S}_\alpha(w_k|x) = \hat{S}_\alpha(w_i|x) < \theta_{i\alpha} \tag{18}$$

where $\hat{S}_\alpha(w_i|x)$ is the estimated similarity measure for class i given x at aspect angle alpha. Turbaugh further extends upon his method by not only evaluating the threshold using the similarity measure of the winning class, but also the difference of similarity values between the winning score and the next closest score. The method labels an exemplar NDEC if the following condition occurs:

$$\max_{k=1,2,\ldots N} \hat{S}_\alpha(w_k|x) - \max_{k=1,2,\ldots N, k\neq i} \hat{S}_\alpha(w_k|x) < \theta_{i\alpha} \quad , \tag{19}$$

where $\theta_{i\alpha}$ is some percentage of the overall range of scores for that exemplar. The equation could also be rewritten as the following:

$$\frac{\max_{k=1,2,\ldots N} \hat{S}_\alpha(w_k|x) - \max_{k=1,2,\ldots N, k\neq i} \hat{S}_\alpha(w_k|x)}{\max_{k=1,2,\ldots N} \hat{S}_\alpha(w_k|x) - \min_{k=1,2,\ldots N} \hat{S}_\alpha(w_k|x)} < \alpha \quad , \tag{20}$$

where the numerator is the difference between the highest score and the second highest score, and the denominator is the range of all the scores; α is the user defined threshold, which range from 0 to 1, where 0 results in no exemplars are given NDEC labels and 1 results in all exemplars are given NDEC labels. For example, in a classification system with 10 target classes, an exemplar receives the similarity metric score S, a 1 x 10 vector:

$$S = (0.25, 0.05, 0.05, 0.05, 0.45, 0.05, 0.05, 0.05, 0.15, 0.1) \quad , \tag{21}$$

where S_5 is identified to be the highest score with a value of 0.45, and therefore, the winning class. The second highest score is also needs to be evaluated, and it is S_1 with a value of 0.25. The difference between S_5 and S_1 is 0.45 - 0.25 = 0.2, the range between the highest score and lowest score is 0.45 - 0.05 = 0.4. The NDEC value would be:

$$\frac{.45 - .25}{.45 - .05} = \frac{.1}{.4} = 0.25 \quad . \tag{22}$$

The exemplar would only receive a NDEC label if α is greater than 0.25.

Turnbaugh's similarity method improves Friend's Entropy and KL methods in the two ways. First, Turnbaugh's method is simple to compute and understand, it simply weights the difference between the two highest scores against the overall range of all the scores. Second, the computation of posterior probability estimate is not required[31].

2.5 Probabilistic Neural Network

Probabilistic Neural Network (PNN) is a neural network with three layers. The input layer, pattern layer and the class or category layer. The input layer consist of d input units, where d is the dimensionality of the data or the number of features. There are n pattern units, one for every training data point used to train the PNN. The category layer consist of class units, one for each target class in the classification system. The input units are fully connected to the pattern units. The pattern units

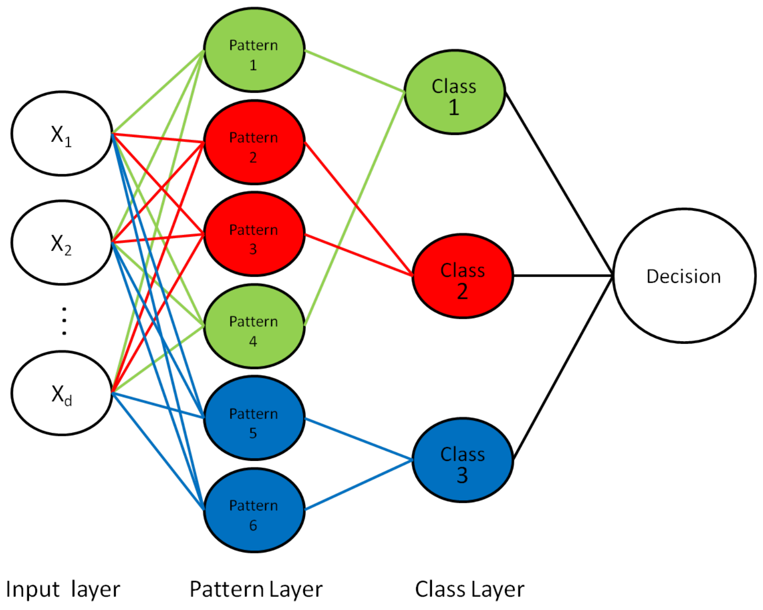


Figure 12. ***Example of a PNN***

are only connected to their corresponding class unit. Figure 12 provides an example of a PNN. PNN is based on the assumption that the data features are independent and identically distributed multivariate normal. During training, each normalized training point is applied to the input units. Each training vector is used to calculate the weights w_k, where $k = 1, 2, ..., n$, each has the value of a component of the vector. In the test phase, the inner product of a test exemplar, x, and each pattern unit is computed to yield the net activation[8]:

$$net_k = w_k^T x \quad . \tag{23}$$

The activation function for each pattern unit is the nonlinear function, ϕ, based on an assumed Gaussian distribution[8]:

$$\phi = e^{-(x^T x + w_k^T - 2x^T w_k)/2\sigma^2} \quad . \tag{24}$$

Or in the normalized case:

$$\phi = e^{(net_k - 1)/\sigma^2} \quad . \tag{25}$$

Where σ is the user defined parameter, also known as the spread value. The class units then sums the activation functions from its associated pattern units and class unit with largest sum is considered the winning target class.

2.6 Fusion

Sensor information fusion is the intelligent integration of data from multiple sources. Dasarathy's model[5] defines three levels of fusion in classification, data level, feature level and decision level. Figure 13 illustrates the Dasarathy input output Fusion model. Data fusion is the merging of information from separate and conceptually different sources and consolidation into one source. Hall[15] states that sensor fusion combines data from multiple sensors, and related information from associated databases, to achieve improved accuracies and more specific inferences than could be achieved by the use of a single sensor alone. Humans and animals use mul-

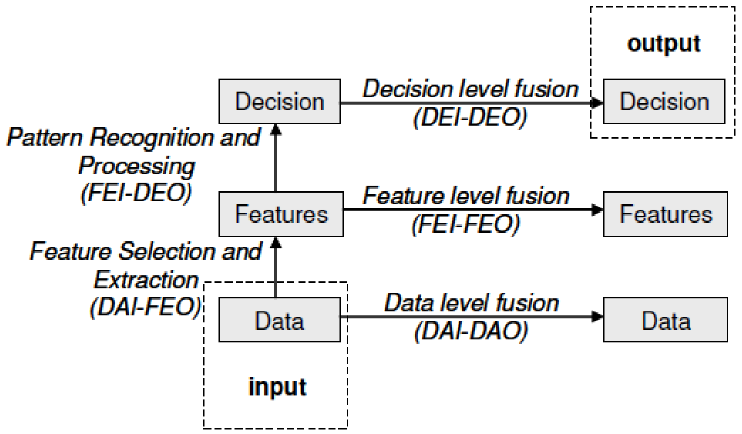

output
Decision
Decision level fusion (DEI-DEO)
Decision
Pattern Recognition and Processing (FEI-DEO)
Features
Feature level fusion (FEI-FEO)
Features
Feature Selection and Extraction (DAI-FEO)
Data
Data level fusion (DAI-DAO)
Data
input

Figure 13. ***Dasarathy I/O Fusion Model, reprinted from [19]***

tisensory data fusion to increase awareness of their surrounding environment. For example, the presence and quality of an edible substance may not be detected solely by sight, but by a combination of sight, touch, smell and taste. In this research, the multisensor data comes from two SAR sensors that operates in different polarities, horizontal(HH) and vertical(VV). The two sensors collect different data from SAR targets, multisensor fusion is employed for classification. Information fusion is not limited to the sensor level, even at the single sensor level; features can be integrated prior to classification. The main goal of this research is the fusion of contextual features with HRR features for classification of targets. Decision level fusion can also be achieved by using multiple classifier systems (MCS). The combination takes advantage of the strengths of the individual classifiers, avoid their weaknesses, and improves classification accuracy[16]. This research combines the decision outputs from a multiple classifiers system containing OOL, NDEC, PNN and template matching classifiers.

2.6.1 MCS Methods.

Ruta and Gabrys [26] state there are three types of methods for classifier fusion. They differ by the type of information produced by the individual classifiers:

1. *Single Class Labels (SCL) methods*: The classifiers from this method output a single discrete class label, providing the least amount of useful information to the fusion process. Voting method is an example of SCL. Consider the following outputs from n classifiers, with m possible classes, as a decision vector d defined as $d = [d_1, d_2, ...d_n]^T$ where $d_i \in \{c_1, c_2, ..., c_m, r\}$, c_i denotes the label of the i^{th} class and r is rejecting the exemplar from any class. The counting function is defined as the following:

$$B_j(c_i) = \begin{cases} 1 & \text{if } d_j = c_i \\ 0 & \text{if } d_j \neq c_i \end{cases} . \tag{26}$$

Then the general voting routine can be defined as following, where $i \in \{1, ..., m\}$:

$$E(d) = \begin{cases} c_i & \text{if } \forall t \in \{1, .., m\} \sum_{j=1}^{n} B_j(c_t) \leq \sum_{j=1}^{n} B_j(c_i) \geq \alpha \cdot m + k(d) \\ r & \text{otherwise} \end{cases} , \tag{27}$$

where alpha is a parameter and $k(d)$ is a function that provides additional voting constraints. The most conservative voting rule is given if $k(d) = 0$ and $\alpha = 1$, meaning that the class is chosen when all classifiers produce the same output. This rule can be liberalized by lowering the parameter. The case where $\alpha = 0.5$ is commonly known as the majority vote.[26]

2. *Class ranking based methods*: Class ranking based methods requires that each classifier produce a list of ranked order possible classes for each exemplar. Ranking based fusion methods combine the rankings produced by all the classifier to produce a overall decision. One example of class ranking based method is known as Borda count. Borda Count is defined as a mapping from a set of individual rankings to a combined ranking leading to the most relevant decision.

The Borda Count for class c_k, $B(c_k)$, is defined as a sum of the number of classes ranked below class c_k by each classifier. The magnitude of the Borda count reflects the level of agreement that the input pattern belongs to the considered class.[26]

3. *Soft-output classifier fusion methods*: The classifiers produce a probability for each class. The outputs, which range from 0 to 1, are also called possibility, necessity, belief or plausibility. Ruta and Gabrys[26] refer to the output values as fuzzy measures. These fusion methods try to reduce the level of uncertainty by maximizing suitable measures of evidence. Bayesian Fusion Methods are examples of Soft-output classifier fusion. Bayesian methods can be used to fuse classifier's output, provided the classifier outputs are expressed in posterior probabilities. The combination of given likelihoods is also a probability of the same type, which is expected to be higher than the probability of the best individual classifier for the correct class.

Two basic Bayesian fusion methods are introduced by Ruta and Gabrys[26]. The first one named Bayes Average is a simple average of posterior probabilities. The second method uses Bayesian Belief Network to provide a belief measure associated with each classifier output and integrates these belief measures into a combined final belief.

2.7 Bayesian Belief Networks

A Bayesian Belief Network, or causal network, is a graphical representation of probability distributions and models the causal relationships between the nodes[2]. The network consists of a set of nodes and a set of directed edges that encodes the dependent relationships among the random variables or events. Each node has a finite set of mutually exclusive states. When two nodes are connected by an edge, the influencing node is called the parent node and the child node is one being influenced. There is also a conditional independence between the variables. For variable X_i, the parent variable is $pa(X_i)$ and X_i is independent of the variable set $A(X_i)$, which is the set of variables that are not children of variable $pa(X_i)$. Then the probability of X is calculated as:

$$P(X_i|A(X_i), pa(X_i)) = P(X_i|pa(X_i)). \tag{28}$$

There is also a conditional probabilities table (CPT) associated with each of the nodes. The table values are defined as the probabilities of the node, given its parent nodes:

$$P(X_i|pa(X_i)). \tag{29}$$

Nodes without parents simply have the prior probability distributions. The Bayesian network can represent the joint probability by utilizing the node relationships and the conditional probability table. Consider the network with variables $X = X_1, X_2, ..., X_n$. Applying the conditional independence into the chain rule, resulting posterior probabilities for X is the following expression:

$$P(X_1, X_2, ..., X_n) = \prod_{i=1}^{n} P(X_i|pa(X_i)). \tag{30}$$

For example, consider the following case in figure 14. Jimmy has a test next week in Math, English or History. The level of difficulty to Jimmy is dependent on what the subject is. Whether he studies or not, also depends on the subject. The results of the test depends on the difficulty and if Jimmy studied or not. *What is the probability that he will pass the test* $(d1)$*?*

Using the Bayesian Net defined in figure 14, node A is the subject, B is the level of difficulty, C is the amount of study, and D is the resulting grade. The lower case variables represent the different outcomes of an event. The tables next to the nodes

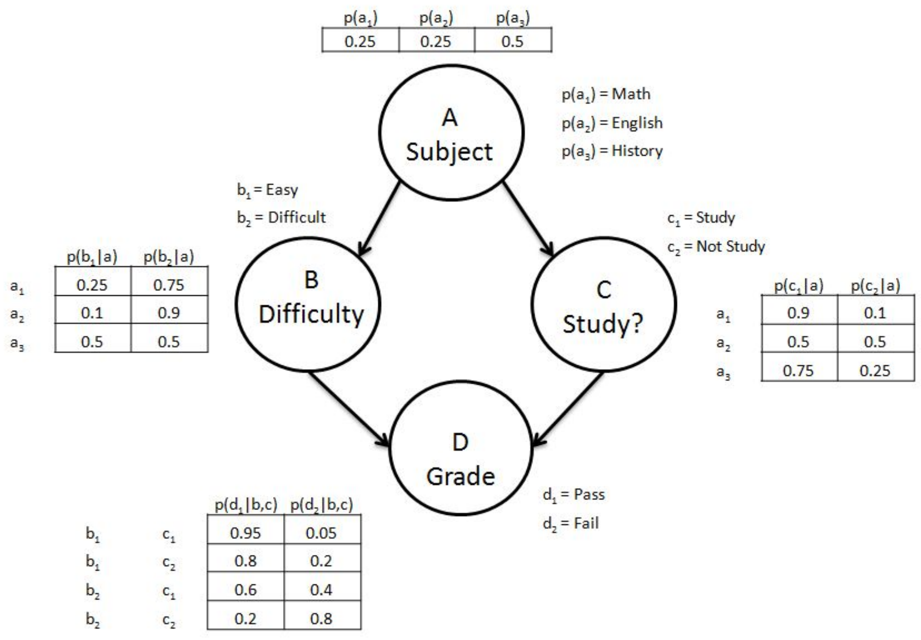


Figure 14. ***A Bayesian Belief Network***

are the CPTs, they provide conditional probabilities for an given event. For example, given that the subject of test is math a_1, then conditional probability for the difficulty being easy b_1 given is $p(b_1|a_1) = .25$, as provided by the CPT. The probability of d_1, Jimmy receives a passing grade can be defined as:

$$P(d_1) = \sum_{a,b,c} P(a,b,c,d_1) \quad , \tag{31}$$

applying the joint probability equation to the above

$$\begin{aligned} P(d_1) &= \sum_{a,b,c} P(a)P(b|a)P(c|a)P(d_1|b,c) \\ &= \sum_{a} P(a)P(b|a)P(c|a) \sum_{b,c} P(d_1|b,c) \quad . \end{aligned} \tag{32}$$

After the taking the test the following week, given the evidence that it was an English test (a_2) and Jimmy had in fact studied (c_1), what is the posterior probability he pass *the test d_1?*

$$P(d_1|a_2,c_1) = \frac{P(a_2,c_1,d_1)}{P(a_2,c_1)} = \frac{\sum_{i=1}^{2} P(a_2,b_i,c_1,d_1)}{\sum_{j=1}^{2}\sum_{i=1}^{2} P(a_2,b_i,c_1,d_j)} \quad . \tag{33}$$

The numerator would be:

$$\sum_{i=1}^{2} P(a_2,b_i,c_1,d_1) = \sum_{i=1}^{2} P(d_1|a_2,b_i,c_1)P(c_1|a_2,b_i)P(b_i|a_2)P(a_2) \quad . \tag{34}$$

If the conditional independence rule is applied:

$$\sum_{i=1}^{2} P(a_2,b_i,c_1,d_1) = \sum_{i=1}^{2} P(d_1|b_i,c_1)P(c_1|a_2)P(b_i|a_2)P(a_2) \quad . \tag{35}$$

Factoring out the constant terms:

$$\begin{aligned}\sum_{i=1}^{2} P(a_2,b_i,c_1,d_1) &= P(c_1|a_2)P(a_2)\sum_{i=1}^{2} P(d_1|b_i,c_1)P(b_i|a_2) \qquad (36)\\ &= (.5)(.25)\cdot((.95)(.1)+(.6)(.9))\\ &= .079 \quad .\end{aligned}$$

The denominator is:

$$\begin{aligned}\sum_{j=1}^{2}\sum_{i=1}^{2} P(a_2,b_i,c_1,d_j) &= \sum_{j=1}^{2}\sum_{i=1}^{2} P(d_1|b_i,c_1)P(c_1|a_2)P(b_i|a_2)P(a_2) \qquad (37)\\ &= P(c_1|a_2)P(a_2)\sum_{i=1}^{2} P(d_1|b_i,c_1)P(b_i|a_2)\\ &= (.5)(.25)\cdot((.95)(.1)+(.6)(.9)+(.05)(.1)+(.4)(.9))\\ &= .125 \quad .\end{aligned}$$

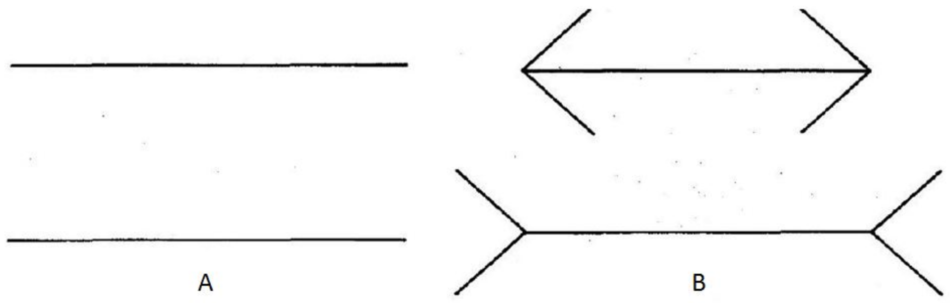


Figure 15. ***Example of Context*****, figure adapted from[30]**

Therefore, the posterior probability Jimmy passed his test (d_1) given the evidence that the test was English (a_2) and he studied (c_1) is:

$$P(d_1|a_2, c_1) = \frac{P(a_2, c_1, d_1)}{P(a_2, c_1)} = \frac{.079}{.125} = .635 \ . \tag{38}$$

2.8 Contextual Information

In pattern recognition, it may be difficult to classify an object when considered out of context. An object may resemble multiple classes, risking the chance of misclassification. An object viewed in isolation may differ when viewed in some context. Toussaint[30] states that automated pattern recognition should use context in order to solve disambiguation, error-correction, and to filling in gaps caused by information missing or covered by background noise. Toussaint gives an example of how human beings use context at the perceptual level to distinguish objects. In Figure 15 A, it appears that both lines are equal in length. However, when the two lines are view in context of the arrows in Figure 15 B, the lower line appears to be longer. In reality, the two lines are still the same, but distinguishable. In automated classification, viewing in context is the selection of features.

In HSI, spectral features are predominately used for classification. However, information about the shape of objects can add context to the spectra features. Zhang et al's [36] pixel shape index (PSI) effectively creates shape features of HSI images. Their integration of PSI showed notable increase of classification accuracy. Yang et al [34] extends on the work of Zhang by creating a two-stage algorithm with two levels of spatial features. First, PSI based features are combined with spectral features for classification, which are then further refined by fusing with high level spatial neighborhood information of the target classes. The high level spatial neighborhood information provides error-correction capability when the lower level spatial features are similar and can lead to misclassification. Corr et al's [4] work in classification of urban SAR images indicates that using additional derived shape and size features from the data can resolve ambiguities between some classes, and enhance accuracy over using polarmetric and interferometric data alone.

Messer[23] incorporated a number of spatial contextual features in his AutoGAD HSI anomaly detection method and combined all the information using a Bayesian Belief Network to generate more accurate results. Two of these metrics will be used in this research. The aspect ratio of an object, Figure 16, is simply the ratio between its length and width. Messer calculated these two parameters using the first and second principal components.

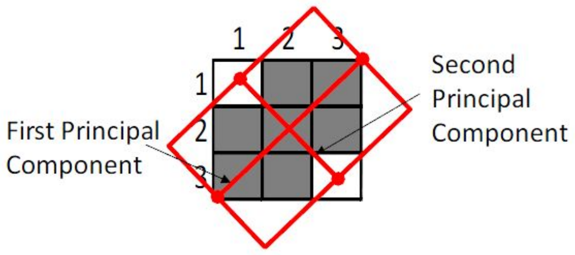


Figure 16. ***Example of Aspect Ratio*****, figure reprinted from[23]**

The Bulbosity of an object, Figure 17, is the ratio between the area defined by the product of its major axis and minor axis and total area of the object image pixels[23].

It relates information about how "filled in" is the rectangle define by the axes

$$Bulbosity = \frac{\text{Major axis} \cdot \text{Minor axis}}{\text{Area}}. \tag{39}$$

Bulbosity can distinguish two different shaped objects with the same aspect ratio. Along with area and mean intensity, Messer uses four contextual values to classify

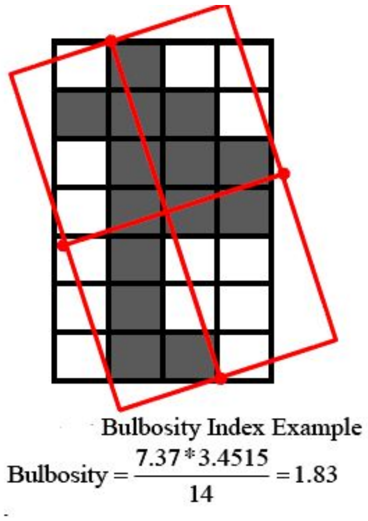


Figure 17. ***Example of Bulbosity*, figure reprinted from[23]**

whether image regions were anomalous or background. During the training phases, the conditional probabilities are computed and utilized within the Bayesian Network during the test phases. Messer's[23] results indicate that the spatial features used significantly improve classification accuracy.

2.9 SAR Image Segmentation

Before contextual feature extraction can be accomplished using a SAR image, we must identify where the object is in the image and which pixels represent the object. The process of segmentation is "recognizing or grouping together the various parts of a composite object"[8]. The object must be segmented from the background. SAR image pixels can be categorized into three distinct regions: target region, shadow region, and clutter region[25]. There many algorithm for SAR image segmentation. This is a review of the most recent and influential on this research.

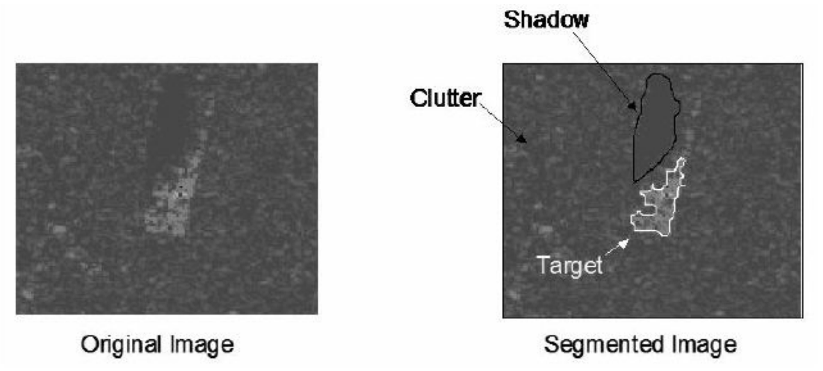


Figure 18. ***Example of a Segmented SAR Image*****, figure reprinted from[25]**

El Zaart et al.[35] uses an algorithm for segmentation of SAR images based on thresholds and the histogram of the image intensities. They used the maximum likelihood technique to estimate the histogram parameters. Using the histograms, thresholds for segmenting the object are established. Kuiying et al. [18] introduces a multi-step process in which the SAR images are pre-transformed to reduce the background noise and clutter prior to segmentation. Using a power transformation and power transformation plus partial differential de-noising, the background noises are significantly reduced. Furthermore, the transformation alters the distribution of the the image intensity from a near Rayleigh distribution to a more normal distributions, which mean and variance parameters can be used as segmentation thresholds. In Nicoli and Anagnostopoulos'[25] work, the segmentation process is achieved using two basic steps that were simple and computationally efficient. The first established a single threshold, where 10% the highest intensity pixel is selected to produce an approximated object's region. The second step is a morphological close operation, where each segmented region is discarded if it does not contain a pixel with intensity higher than 40% of the highest overall pixel intensity. A higher threshold is re-established and the close operation repeats until only one target region remains. The remaining region is the segmented object.

III. Methodology

This research investigates the potential to improve template classification of SAR targets through the use of contextual features. Along with contextual information, we also apply a Bayesian Belief Network process to fuse output information from template matching, OOL, and NDEC algorithms to generate an improved overall classification. This section outlines and summarizes the data processing steps, methods of feature extraction and algorithms used in classification. The measures of performance that are used to evaluate this methodology are also defined.

3.1 Feature Extraction methods

Prior to any classification, the data features must be defined and organized. Figure 19 displays the data organization process for this research. The SAR data processing begins with two parallel processes, HRR features extraction and contextual features extraction. HRR feature extractions follow the process described Section 2.2.1.

The exemplars with its associated HRR and contextual features are separated into either a training data set or the test data set, depending upon each data points actual depression angle and the purpose of the experiment. In Normal Operating Conditions (NOC) experiments, the system is tested using data with the same range of depression angles as the data used in training. In Extended Operating Conditions (EOC) experiments, the system is tested using data with a wider range of depression angles than the data used to train the classification system. The result is that the system has to classify objects from angles of view it has not trained on. Table 3 shows the data split scheme used for experiments. EOC1 splits the data below 8 degrees into training and testing, while all data above 8 degrees are testing. EOC2 has a mutually exclusive set of depression angle data for training and test.

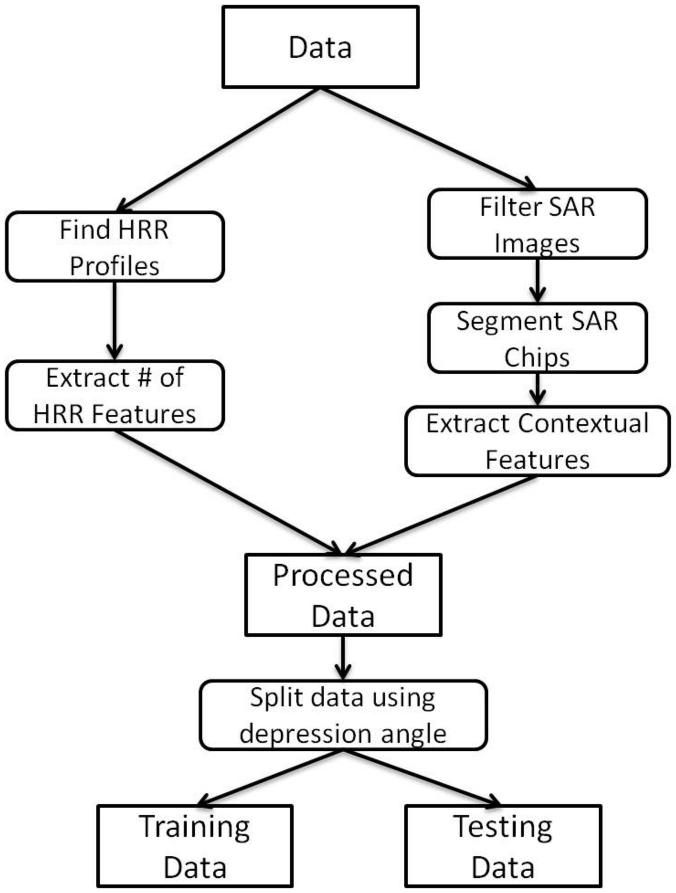


Figure 19. ***Data Organization Process***

Data Split Scheme	**Depression Angle**			
Operating Conditions	$<= 8°$		$> 8°$	
Normal Operating Condition (NOC)	50% Train	50 % Test	50% Train	50 % Test
Extended Operating Condition 1 (EOC1)	50% Train	50 % Test	Test	
Extended Operating Condition 2 (EOC2)	Train		Test	

Table 3. Data Split Scheme

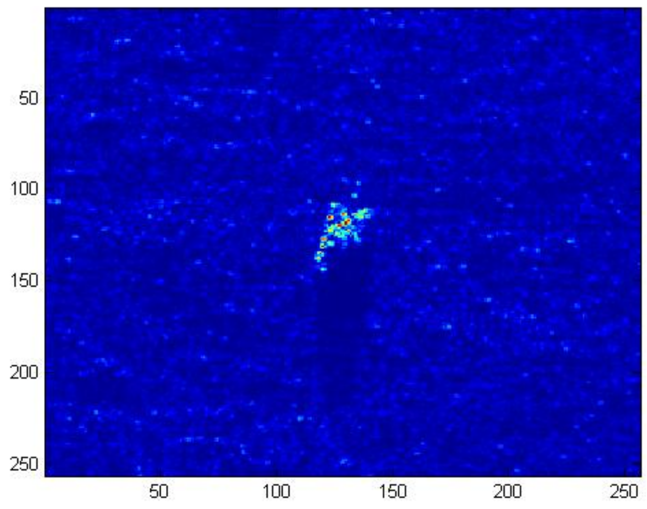


Figure 20. ***Example of a SAR image chip***

3.2 Segmentation of SAR chips

SAR chips used for this research are 256 by 256 pixel images with the signal intensities being the magnitude of each pixel. The target objects are centered in the image with only one target object per image. Figure 20 below is example of a T-72 tank's SAR image chip. Colors reflect intensity, highest intensity pixels are red and blue pixels represent low intensities, assumed to be mostly background clutter. Note that all example of SAR images in this chapters are displayed inverted with the sensor at the top of the image and the shadows shown below the objects.

3.2.1 Filter algorithms.

The SAR images are transformed prior to segmentation to reduce the amount of background clutter. The function used is based off the pixel's distance from the center. Since we assume that the objects of interest are located at the center of each SAR image, this research transforms the image intensities according to its relationship from the center of the object. The further a pixel is from an object's center, the less

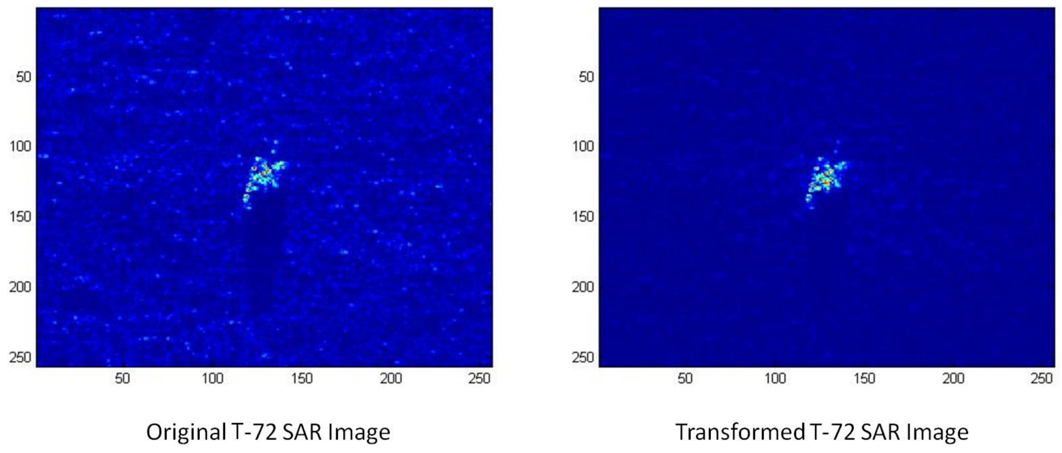


Figure 21. ***Example of Image Transformation***

likely that it is an object pixel. Given that a SAR image is organized as an m x n matrix with intensity values, each element of the matrix represents a pixel. Each pixels distance from the center is defined by its location on the row i and column j and the following function:

$$dist_{ij} = \sqrt{\left(i - \frac{m}{2}\right)^2 + \left(j - \frac{n}{2}\right)^2}. \tag{40}$$

The image intensity is than transformed using the following function:

$$Transformed\, Intensity_{ij} = \frac{Intensity_{ij}}{\sqrt{dist_{ij}}}. \tag{41}$$

Figure 21 shows an example of the transformation process. The image on the left is the original SAR image of a T-72 tank, notice all the background clutter all around the image. The image on the right is the same image post transformation. The background clutter has been significantly reduced and the target object stands out more.

3.2.2 Segmentation.

The first step in the segmentation process is to filter out candidate pixels from the transformed image that may contain the target object. A binary image is created based on the distribution of pixel intensity. Pixels with intensities greater than 2 standard deviation above the image mean intensity are designated candidate points with a binary value of one,

$$candidate\,pixels > \mu + 2\sigma \quad . \tag{42}$$

All other pixels are considered to be background and are given a value of zero. Mean and standard deviation are calculated using the following equations:

$$\mu = \frac{\sum_{i=1}^{256} intensity_{ij}}{total\,number\,of\,pixels} \tag{43}$$

$$\sigma = \sqrt{\frac{\sum_{i=1}^{256}\sum_{i=1}^{256}(intensity_{ij} - \mu)^2}{total\,number\,of\,pixels}} \quad . \tag{44}$$

Figure 22 shows the binary image of the T-72 tank after the threshold has been applied.

The second step reevaluates a pixel's candidacy in context of its 8 surrounding pixels. A candidate pixel is confirmed to be a target pixel only if N or more of its 8 surrounding pixels are also candidate pixels. N is defaulted to 6, based on a visual evaluation of segmentation performance on random samples of in library target images. The pixels that pass both segmentation steps are labeled as target pixels and the resulting binary images are considered segmented. Figure 23 shows the final binary image after the segmentation process of the T-72 tank image, the black pixels have values of zero and white pixels have values of one.

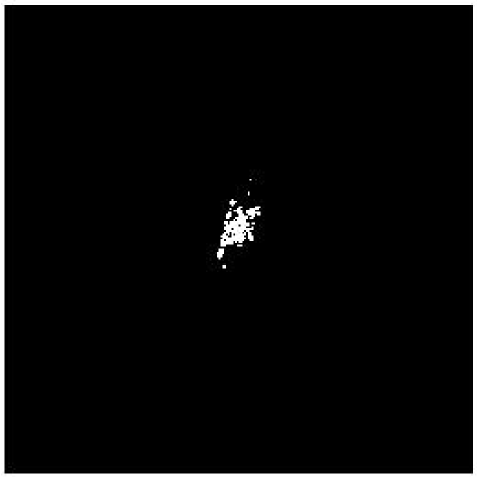


Figure 22. ***Binary Image of T-72 after first segmentation step***

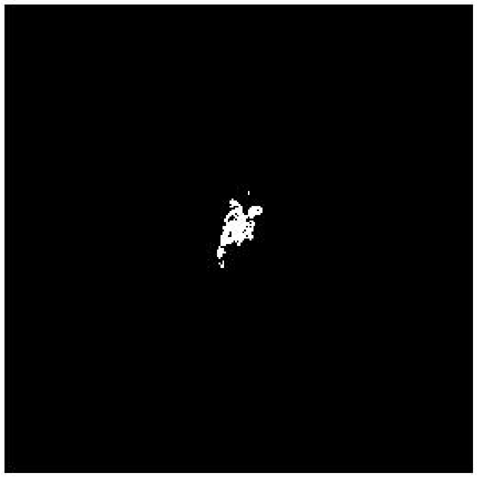


Figure 23. ***Final Segmented Binary Image of T-72p***

3.3 Context Extraction

After a SAR image has been segmented, the contextual information can be extracted from the binary image. This research examines three different methods to extract a target object's aspect ratio and one method to determine the target's bulbosity index.

3.3.1 RegionProps Aspect Ratio Method (AR1 Method).

Aspect Ratio is a ratio relationship between an object's length and width. Square objects will have an aspect ratio of 1, and long objects will have an aspect ratio greater than 1. Note that no object will have an aspect ratio value of less than 1, since the longer measure will always be considered the length. This method uses MATLAB function regionprops[22], previously used in Messer's[23] work in HSI to extract the aspect ratio from a binary image. Regionprops uses the multiple disconnected shapes or some cases, the single shape of a binary image, to create a convex polygon of the object. RegionProps obtains the aspect ratio by finding the Major Axis and Minor Axis of the convex polygon. Figure 24 displays an example of the RegionProps aspect ratio extraction method on the previously segmented T-72 binary image. AR1 is defined by:

$$AR1 = \frac{Major\ Axis\ Length}{Minor\ Axis\ Length} = \frac{34.7}{17.9} = 1.938\ \ . \tag{45}$$

Using the Regionprops method, we find the T-72 tank at this particular aspect angle and depression angle has an aspect ratio of 1.938.

3.3.2 Principal Component Aspect Ratio Method (AR2 Method).

This approach uses Principal Component Analysis (PCA) to extract the aspect ratio information from the original SAR images. We first standardize the intensity

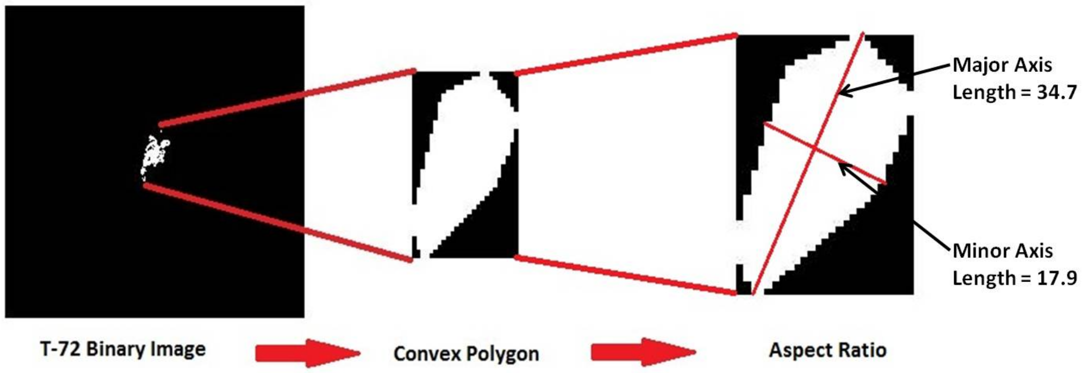


Figure 24. ***RegionProps Aspect Ratio Extraction method***

values of the SAR image matrix and find the correlations matrix. Using MATLAB's princomp function, PCA is performed on the correlations matrix, the eigenvalues and eigenvectors are obtained. The two largest eigenvalues represent the magnitude of the first two principal components. Since the first principal component accounts for the largest amount of the total variation in image intensity, its corresponding eigenvalue should be the length of the object. The second principal component accounts for the maximum amount of the remaining total variation not explained by the first principal component. The eigenvalue corresponding to the second principal component should be the image object's width. Therefore, aspect ratio can calculated as the ratio of the largest eigenvalue over the second largest eigenvalue. Using the original T-72 SAR image, this has 256 by 256 pixel dimensions, PCA results in a 256 eigenvalue vector. The largest eigenvalue is 15.1 and the second largest is 11.4. AR2 is defined by:

$$AR2 = \frac{Largest\ Eigenvalue}{Second\ Largest\ Eigenvalue} = \frac{15.1}{11.4} = 1.324\ . \tag{46}$$

Using the PCA method, we find the T-72 tank at this particular aspect angle and depression angle has an aspect ratio of 1.324.

3.3.3 Mixed Aspect Ratio Method (AR3 Method).

This method combined approaches from AR1 method and AR2 method. The images are transformed like in AR1 Method; Contextual information is extracted by performing PCA on the transformed SAR images instead of the original. AR3 will also be the ratio between the largest eigenvalue and the second largest eigenvalue. AR 3 is defined by:

$$AR3 = \frac{Largest\ Eigenvalue}{Second\ Largest\ Eigenvalue} = \frac{26.37}{12.2} = 2.157 \ . \tag{47}$$

Using the AR3, we find the T-72 tank at this particular aspect angle and depression angle has an aspect ratio of 2.157

3.3.4 Bulbosity Method.

The Bulbosity index is additional contextual feature we can extract from SAR images. This index can potentially provide distinction for objects with similar HRR profiles and aspect ratios, but have different surface areas. The Bulbosity index method used in this research is a simple extension of Messer's [23] HSI approach SAR images. Recall that the segmentation process created binary images of our target objects. We will use the total number of pixels segmented as the estimation of the object's surface area corresponding to its aspect angle and depression angle of view. The MATLAB function regionprops will provide the major and minor axes lengths, and the Bulbosity index is defined by the following:

$$Bulbosity\ Index = \frac{Major\ Axis\ Length \cdot Minor\ Axis\ Length}{Object\ Surface\ Area} \ . \tag{48}$$

Using the T-72 segmented binary image, we can extract its Bulbosity index. Figure 25 shows the T-72 tank Bulbosity index extraction process. The total number of

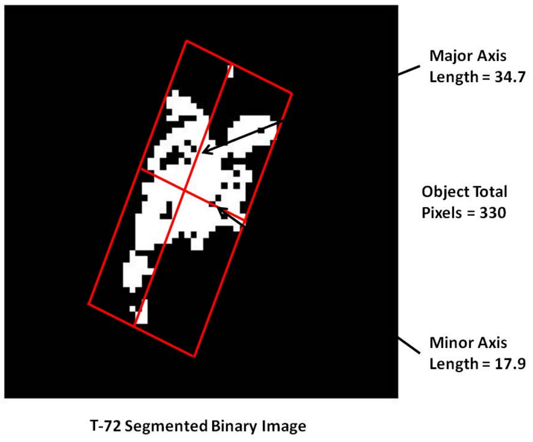


Figure 25. ***Bulbosity index extraction process***

target pixels in the image is 330, which becomes the estimate for the surface area of the tank. The example T-72 image's Bulbosity index can be calcuated as:

$$\begin{aligned} Bulbosity\ Index &= \frac{Major\ Axis\ Length \cdot Minor\ Axis\ Length}{Object\ Surface\ Area} \\ &= \frac{3.47 \cdot 17.9}{330} = 1.88 \ . \end{aligned} \tag{49}$$

Using this Bulbosity method, the Bulbosity index of the T-72 tank at this particular aspect angle and depression angle is 1.88.

3.4 Classification

In order for any supervised classifier to identify an object, it must be provided examples of the discriminating features of the objects in each class. This process is referred to as training. Figure 26 is an illustration of the entire classifier training process for the template classification system used in this research.

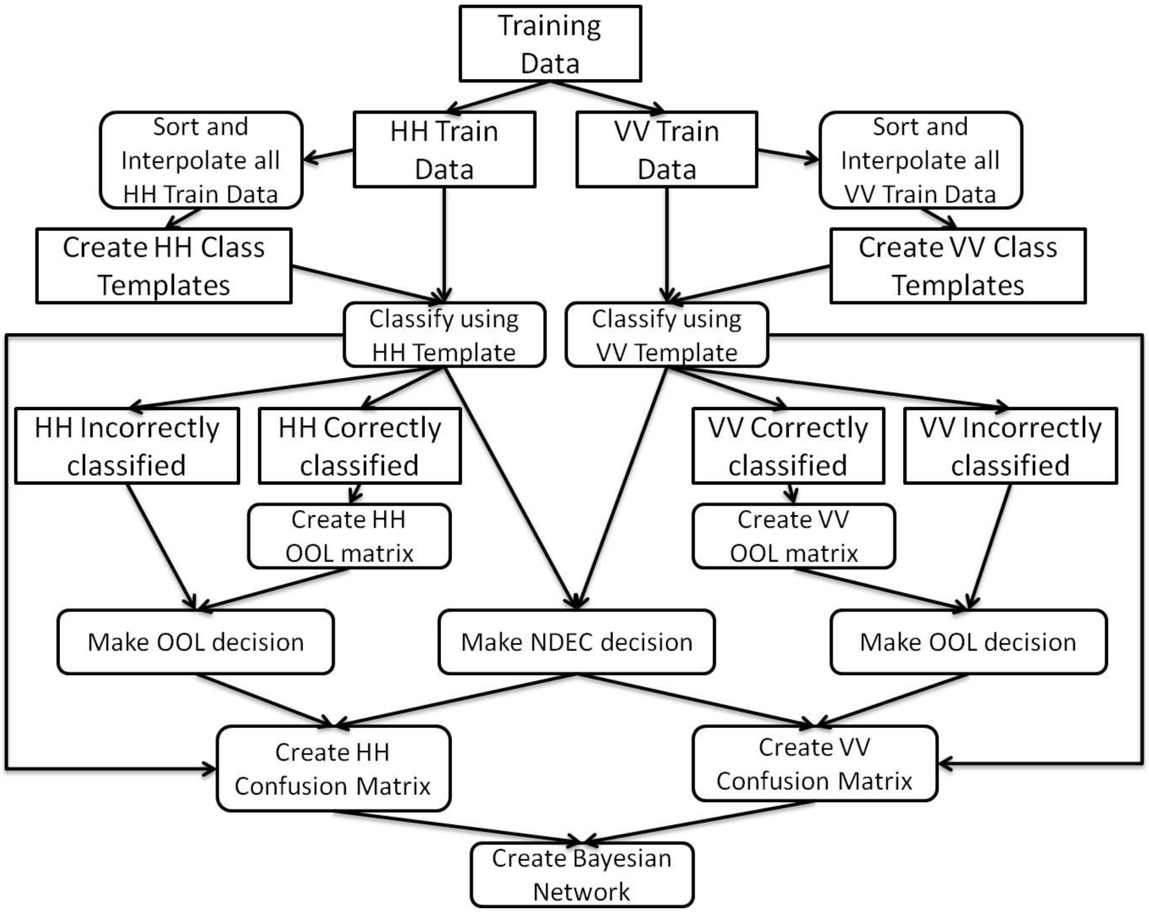

Training Data
HH Train Data
VV Train Data
Sort and Interpolate all HH Train Data
Sort and Interpolate all VV Train Data
Create HH Class Templates
Create VV Class Templates
Classify using HH Template
Classify using VV Template
HH Incorrectly classified
HH Correctly classified
VV Correctly classified
VV Incorrectly classified
Create HH OOL matrix
Create VV OOL matrix
Make OOL decision
Make NDEC decision
Make OOL decision
Create HH Confusion Matrix
Create VV Confusion Matrix
Create Bayesian Network

Figure 26. ***Classifier Training Process***

Table 4. Target Classes and Categories

Friendly	Enemy	Out of Library
Zil-131 med	SCUD	SA-8 TZM
HMMWV	Smerch	BMP-1 tank w/small turret
M113	SA-6 Radar	BTR-70 8 wheeled APC
Zil-131 small	T-72	SA-13 Turreted SAM
M35 Large	SA-6 Tel	SA-8 Tel

3.4.1 Creating Templates.

Each SAR look at an object is recorded in the two polarities, horizontal (HH) and vertical (VV). Since each radar polarization has slightly different returns, we may treat each polarity as a separate sensor and classify each exemplar's two sets of features separately according to polarity. As shown in Figure 26, the result is that for each target class, two templates are created, one for each polarity. All the training data are organized by its true class; Table 4 categorized the 15 target classes. The 10 classes in the friendly and enemy columns are considered in library, we need to create templates for them. Data not belonging to the 10 library class are out of library; they represent the unknown objects on the battlefield, which we have no data on. In the real world, out of library objects have no library templates; hence we do not create templates for out of library target. Within each of the 10 in library classes, data groups are sorted by aspect angle. The data features are interpolated for 360 degrees. Templates are created for each 15 degree wedge; hence each class will have two sets of 24 templates, one for each polarization.

All of the training data are then processed and matched against the 10 sets of library templates to evaluate system performance and provided information for OOL, NDEC, and Bayesian methods. As previously explained in Section 2.4, by we assume the aspect angle of each data point are within $\pm 22.5°$ of the true angle. During training and testing, each exemplar is compared to three 15 degree templates for every target class in the target library. The first wedge is the wedge that contains

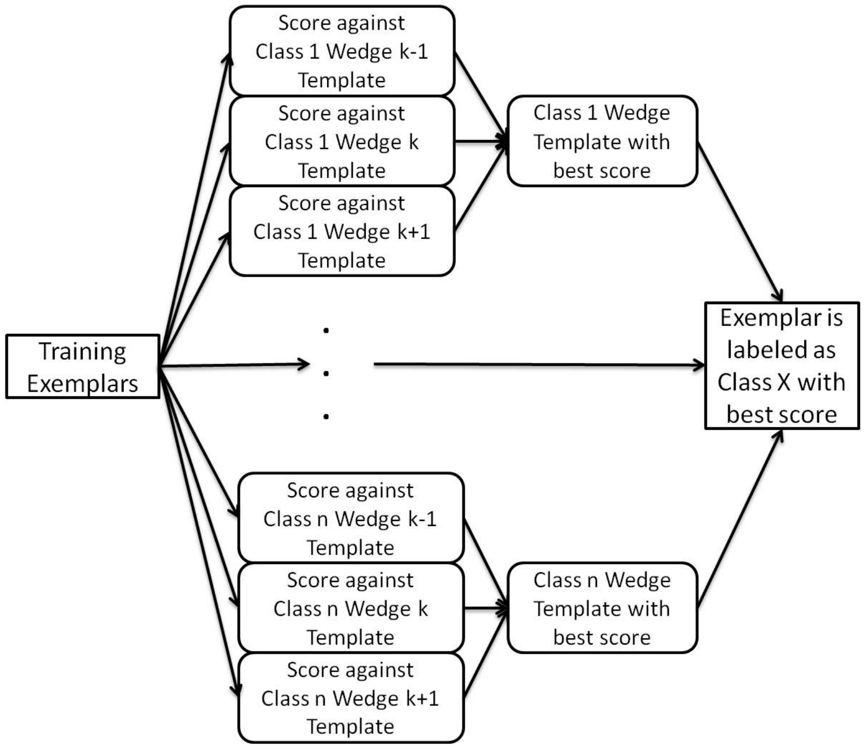

Score against Class 1 Wedge k-1 Template
Score against Class 1 Wedge k Template
Score against Class 1 Wedge k+1 Template
Class 1 Wedge Template with best score
Training Exemplars
Exemplar is labeled as Class X with best score
Score against Class n Wedge k-1 Template
Score against Class n Wedge k Template
Score against Class n Wedge k+1 Template
Class n Wedge Template with best score

Figure 27. ***Wedge Template Matching Process***

the data point's assumed aspect angle, and the wedges to the left and right of the assumed angle are also used to match our aspect angle knowledge assumptions. For example, a data point collected at 14 degrees aspect angle will be matched against template 1 ($0° - 15°$), template 2 ($15° - 30°$), and template ($345° - 360°$). The wedge template with shortest Mahalanobis distance of the three will represent the class against the other library classes. The class with the overall shortest Mahalanobis distance or lowest determinate score is considered the winning classes, or the class the data point will initially labeled as. Figure 27 demonstrates the wedge template matching process.

3.4.2 OOL matrix and NDEC methods.

The OOL matrix method used in this research is similar to Friend[13] and Turnbaugh's[31] quartile methods. To begin the process, all correctly classified training exemplars are separated into the 10 library classes and into the wedge of the true aspect angle it belongs in. The result of this process is that each library class has not one OOL threshold, but 24 OOL thresholds, one for each wedge template. During testing, an exemplar's winning class's Mahalanobis distance is compared to the threshold determined in training. Any exemplars whose Mahalanobis distance is greater than the OOL threshold is labeled OOL. The classification system used for this research employs Turnbaugh's[31] NDEC algorithm, discussed in section 2.4.2, to make non-declaration decisions. However, this method is applied using posteriors probabilities, instead of similarity metric scores.

3.4.3 PNN Classification.

Similar to template classification, the training data set is organized and separated into the 24 wedges according to aspect angle. Using the 24 separate data sets, a PNN is created and trained for each 15 degree wedge. During the test phase, the same aspect angle assumptions are applied. Each test exemplar is classified using the trained PNN from its assumed wedge and the two adjacent wedge PNNs. PNN classification is decided with a majority vote of the outputs from the three PNN classifiers. In the case of a 3 way tie, the class associated with the pattern unit that has the highest activation function value, defined by Equation 24, from any of 3 PNNs is considered is the winning class. Figure 28 illustrates the PNN classification process used in this research.

3.4.4 Bayesian Belief Network.

Each classification algorithm has its strengths and weakness; some algorithms are better at identifying types of class or identifying objects at certain aspect angles than other algorithms. In previous works, not all the system algorithms were used for each

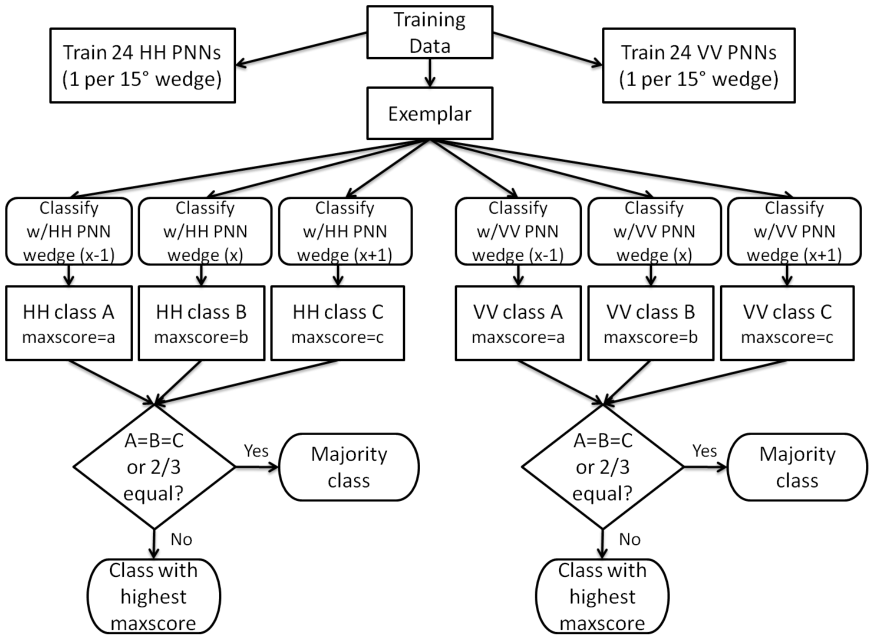

Training Data
Train 24 HH PNNs (1 per 15° wedge)
Train 24 VV PNNs (1 per 15° wedge)
Exemplar
Classify w/HH PNN wedge (x-1)
Classify w/HH PNN wedge (x)
Classify w/HH PNN wedge (x+1)
Classify w/VV PNN wedge (x-1)
Classify w/VV PNN wedge (x)
Classify w/VV PNN wedge (x+1)
HH class A maxscore=a
HH class B maxscore=b
HH class C maxscore=c
VV class A maxscore=a
VV class B maxscore=b
VV class C maxscore=c
A=B=C or 2/3 equal?
Yes
Majority class
No
Class with highest maxscore
A=B=C or 2/3 equal?
Yes
Majority class
No
Class with highest maxscore

Figure 28. ***PNN Classification Process***

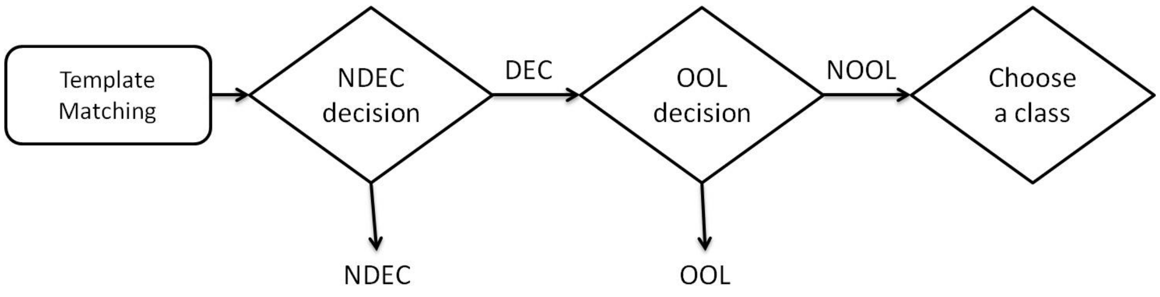


Figure 29. ***Previous decision chains***

exemplar. The decisions chain in Figure 29 describes the issue. For example, if an exemplar receives a NDEC label (or non declaration), the exemplar is not considered for OOL decision. In order to maximize the amount of information in this classification system, template matching, OOL, and NDEC algorithms will be applied to all exemplar. However, if all algorithms are applied regardless of other parallel decision, the previous decision hierarchy cannot be used, since the decisions are made in series. The information from all three sources can be fused together for an overall decision using a Bayesian Belief Network. Figure 30 shows diagram for the new Bayesian Belief Network decision process. Using this Bayesian network, the posterior probabil-

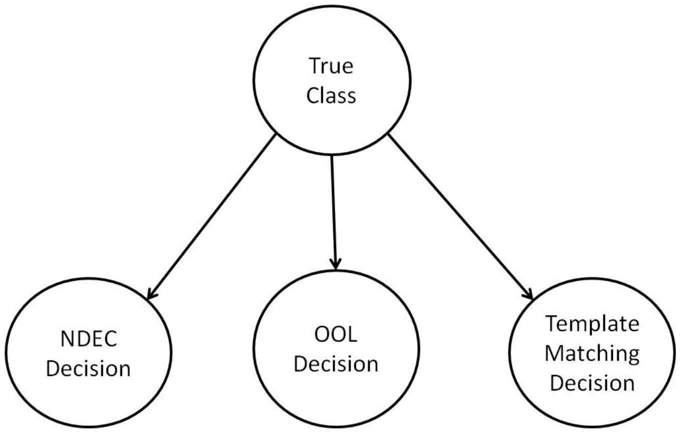


Figure 30. ***New Bayesian Network***

ities for class i can be calculated using equation X. Since there are 10 library classes, there are 11 total true classes when OOL class is counted. The posterior probability will be a vector of 11 probabilities. MATLAB BNT package is used for calculating Bayesian posterior probabilities in this research[24]. The new posterior probabilities can be calculated as:

$$P(Class_i|NDEC, OOL, Temp) = \frac{P(NDEC, OOL, Temp|Class_i)}{\sum_{j=1}^{11} P(NDEC|Class_j) \cdot P(OOL|Class_j) \cdot P(Temp|Class_j) \cdot P(class_j)} . \quad (50)$$

A further extension this Bayesian network is the additional of nodes containing contextual information of the true classes, illustrated in Figure 31. The continuous ranges of aspect ratios and bulbosity index are separated into discrete bins. From the training results, we can calculate the conditional probabilities being in one of those given one of the in library classes.

Since we are treating the two SAR polarities are separate sensors, we need to combine the classification results into one final decision. The Bayesian network is used to complete this final step. The dependent nodes in the network are separated two sets of node, representing the two polarities. Figure 32 illustrates the full Bayesian

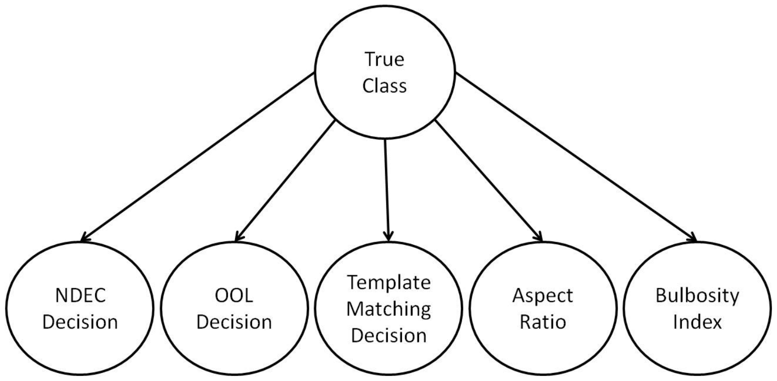


Figure 31. ***Extended Bayesian Network***

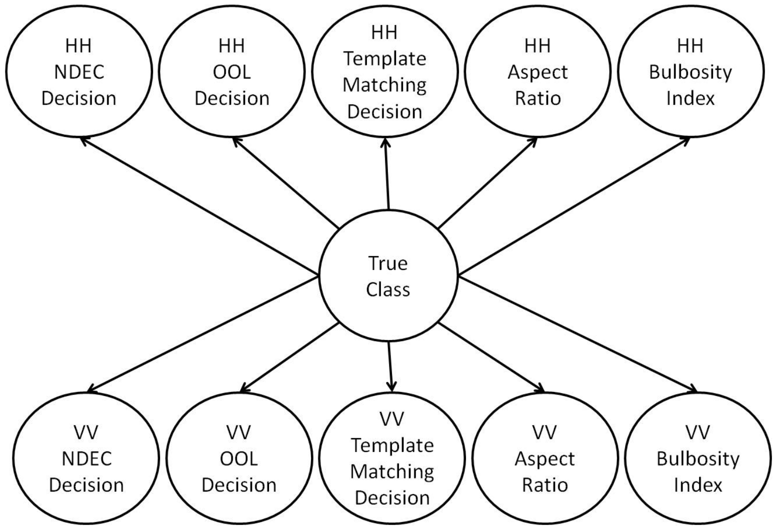


Figure 32. ***Full Bayesian Network***

Belief Network used in this research; all nodes are dependent on the truth class node.

3.4.5 Multiple thresholds for OOL and NDEC.

When NDEC or OOL decisions are made, only binary information is outputted and the information used to make those decision are lost in the decision chain. No level of confidence can be applied to those decisions. One approach from Messer's [23] work was to apply multiple thresholds for each of the decision algorithms. This research also applied this method to NDEC and OOL decisions. NDEC produces

	OOL Threshold 1	OOL Threshold 2	OOL Threshold 3	NOOL
Class 1	15	97	57	76
Class 2	56	53	41	58
Class 3	38	34	81	45
Class 4	18	35	4	63
Class 5	6	58	22	65
OOL Class	102	67	89	10

Table 5. Example of Training OOL Distribution

either a class decision or one of the multiple levels of NDEC threshold breached by the exemplar, giving the decision a level of quasi-confidence. The OOL decision also uses this method to produce one of multiple possible results. These decisions produced with some level of confidence provides a higher level fidelity of evidence for the Bayesian network.

3.4.6 Conditional Probability Tables.

As discussed in Section 2.7, an essential part training a Bayesian Belief Network is gathering the necessary information to populate the conditional probability tables of the network. Tables 5 and 6 provide an example how conditional probability tables are populated. Table 5 is an example of the distribution of training exemplars according to classes and its OOL decision in training.

There are 5 classes and 1 OOL class in this example, represented by the 6 rows and 3 OOL thresholds, represent by the 4 columns that includes not out of library (NOOL). Since the OOL threshold are ascending order, an exemplar's OOL label will be the highest threshold its Mahalanobis distance exceeds, and is labeled NOOL if it is below Threshold 1. In the example, class 1 had 15 exemplars that exceed OOL Threshold 1. To find the conditional probability that an exemplar will exceed OOL threshold given it is from class 1, simply divide event's occurrence by the total occurrences of class 1, which is summed across the first row; as in equation 51

$$P(OOL_1|Class_1) = \frac{OOL_1Class_1}{(OOL_1Class_1) + (OOL_2Class_1) + (OOL_3Class_1) + (NOOL\,Class_1)}$$
$$= \frac{15}{15 + 97 + 57 + 76} = .061 \quad . \tag{51}$$

The process is repeated for the entire population to create conditional probability table for the OOL node, as shown in table 6. The trained Bayesian belief network is

	OOL Threshold 1	OOL Threshold 2	OOL Threshold 3	NOOL
Class 1	0.061	0.396	0.233	0.310
Class 2	0.269	0.255	0.197	0.279
Class 3	0.192	0.172	0.409	0.227
Class 4	0.150	0.292	0.033	0.525
Class 5	0.040	0.384	0.146	0.430
OOL Class	0.381	0.250	0.332	0.037

Table 6. Example of a populated Conditional Probabilities Table

then used in test to combine the all the separate decisions of the OOL, NDEC, and Template matching classifiers and fuse them into one final decision with improved accuracy, as shown Figure 33.

3.5 Evaluations measures

3.5.1 Aggregate measures.

The aggregated true positive rates of identified targets are organized by the three categories: Friendly, Enemy, or Out of Library[13]. Table 4 shows the three categories. If an exemplar is classified as one of the classes from its category, then it will be considered an aggregated true positive. For example, if a SCUD exemplar is classified as a T-72, then it will be considered an enemy. Therefore, the result is an aggregated

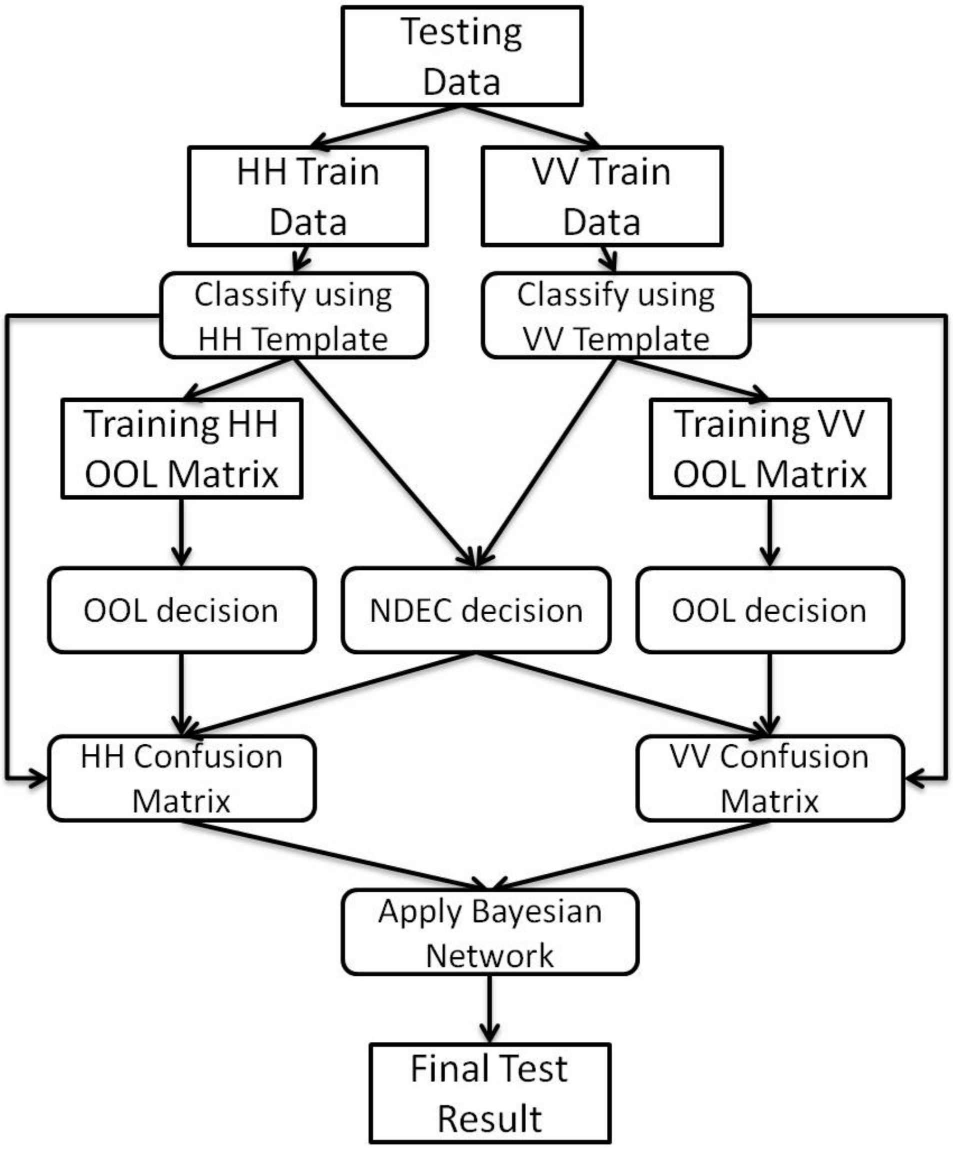


Figure 33. ***Testing process***

enemy true positive. However, if the same SCUD is classified as a M113, then will be consider a friendly object and resulting in an aggregated enemy false negative.

Aggregated Enemy True Positive rate (Agg ETP): This is the classification accuracy rate of the system for identifying enemy targets at the aggregated level. This aggregated measure is not concerned with identifying the exemplars exact class of enemy, as long as it is correctly identified as an enemy target. Exemplars resulting in non-declaration decision are not be considered in the measure

$$AggETP = \frac{Total\, number\, of\, enemy\, exemplars\, identified\, as\, enemy\, targets}{Total\, number\, of\, enemy\, exemplars\, with\, declarations} \quad . \tag{52}$$

Aggregated Friendly True Positive rate (Agg FTP): This is the classification accuracy rate of the system for identifying friendly targets at the aggregated level. This aggregated measure is not concerned with identifying the exemplars exact class of friend, as long as it is correctly identified as a friendly target. Exemplars resulting in non-declaration decision will not be considered in the measure

$$AggFTP = \frac{Total\, number\, of\, friendly\, exemplars\, identified\, as\, friendly\, targets}{Total\, number\, of\, friendly\, exemplars\, with\, declarations} \quad . \tag{53}$$

Aggregated Classification Accuracy rate (Agg TCA): This is the classification accuracy rate of the system for identifying all three categories: Enemy, Friendly, and Out of Library. This aggregated measure is not concerned with identifying the exemplars exact class, as long as it is correctly identified by its category. Exemplars resulting in non-declaration decision are not be considered in the measure

$$AggCA = \frac{Total\, number\, of\, in\, library\, exemplars\, correctly\, identified\, by\, category}{Total\, number\, of\, in\, library\, exemplars\, with\, declarations} \quad . \tag{54}$$

3.5.2 Class level accuracy measures.

The class true positive rates are measures on how well the system can identify exemplars at the class level. In order for a result to be considered a class true positive, the system must identify the exemplar by its true class. For example, a SCUD exemplar classified as a T-72 will result in a class false negative. Only if that SCUD exemplar is classified as a SCUD will be considered a class true positive.

Enemy True Positive rate (ETP): This is the classification accuracy rate of the system for identifying enemy targets at the class level. This measure is concerned with identifying the enemy exemplars exact class of enemy, only classification as the exemplars true class are counted as correct. Exemplars resulting in non-declaration decision are not considered in the measure

$$ETP = \frac{Total\,number\,of\,enemy\,exemplars\,correctly\,identified\,by\,class}{Total\,number\,of\,enemy\,exemplars\,with\,declarations}\quad . \tag{55}$$

Friendly True Positive rate (FTP): This is the classification accuracy rate of the system for identifying friendly targets at the class level. This measure is concerned with identifying the friendly exemplars exact class, only classification as the exemplars true class are be counted as correct. Exemplars resulting in non-declaration decision are not considered in the measure

$$FTP = \frac{Total\,number\,of\,friendly\,exemplars\,correctly\,identified\,by\,class}{Total\,number\,of\,friendly\,exemplars\,with\,declarations}\quad . \tag{56}$$

Classification Accuracy rate (CA): This the total classification accuracy rate of the system at identifying all the target classes and out of library targets. This class level measure is concerned with identifying the exemplars exact class, or if it is out of library. Only classifications as the exemplars true class or correctly identifying an out of library exemplar are counted as correct. Exemplars resulting in non-declaration decision are not considered in the measure

$$CA = \frac{Total\,number\,of\,in\,library\,exemplars\,correctly\,identified\,by\,class}{Total\,number\,of\,in\,library\,exemplars\,with\,declarations}\quad . \tag{57}$$

3.5.3 Classifier level performance measures.

Out of Library rate (OOL rate): This is the rate the OOL classifier correctly identifies objects which are not in the library of templates. This measure evaluates how well an OOL algorithm is performing

$$OOL\,rate = \frac{Total\,number\,of\,exemplars\,correctly\,identified\,as\,OOL}{Total\,number\,of\,OOL\,exemplars}\quad . \tag{58}$$

Declaration rate (Dec Rate): This is the systems rate of making a decision on declaring an exemplar as either a class or out of library. This measure evaluates how well a Non-Declaration algorithm is performing

$$Dec\,rate = \frac{Total\,number\,of\,exemplars\,with\,declarations}{Total\,number\,of\,exemplars}\quad . \tag{59}$$

IV. Results and Analysis

This section analyzes the classifier performance improvements achieved by using the context features created by the 3 aspect ratio methods and Bulbosity methods discussed in Chapter III. Classifier performance from the contextually enhanced systems is compared to the baseline system with only HRR features. The baseline for this research consists of the classifier system with only 10 HRR features. AR1, AR2, AR3 and Bulb designate the performance of the systems with its corresponding contextual feature along with baseline features. Results with label AR1+bulb, AR2+bulb, or AR3+bulb are designated systems with both baseline, aspect ratio and Bulbosity index features included.

4.1 Analysis of Contextual Features

Similar to HRR profiles, the aspect ratio and Bulbosity index of an object will change, depending on aspect angle. Geometry of the object is altered when view from different angles. The four contextual features' averaged values across all classes according to aspect angle are shown in the Figure 34 and 35. Figure 35 shows the systems in NOC experiment, the two sets of different colors indicate the training sets of data and the testing sets of data. In NOC, the two data sets are quite similar, hence the contextual features are also similar. All three of the aspect ratio data indicates a similar behavior at particular aspect angles. As the aspect angles approach 90 degrees and 270 degrees, the aspect ratios increase as sensor views approach the sides of the objects; their shapes appear longer compared to other angles. The opposite is true for Bulbosity index, as the sensors angle of view approach the object's side, the product of the aspect length and width decreases. By equation 48, at those particular angles, the Bulbosity index is at the lowest.

Figure 35 shows the four contextual features' averaged values across all classes

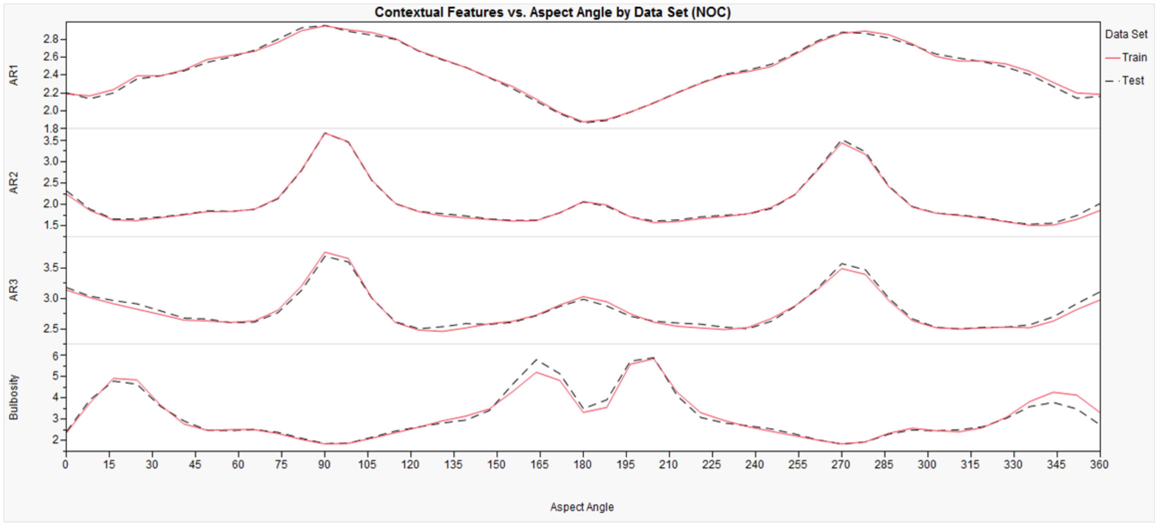


Figure 34. ***Contextual Features by Aspect Angle by Data Set in NOC***

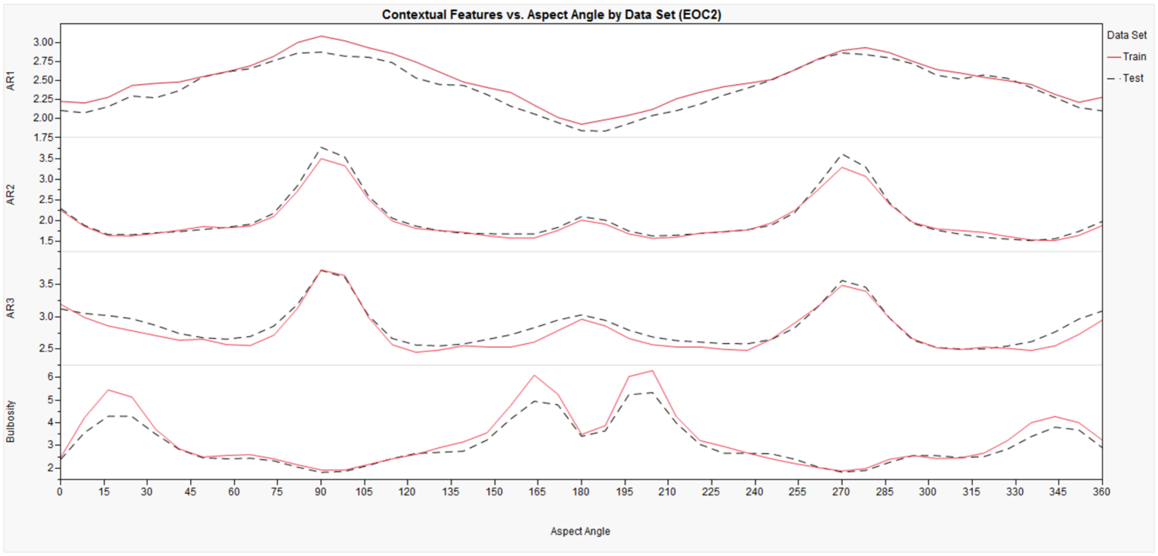


Figure 35. ***Contextual Features by Aspect Angle by Data Set in EOC2***

according to aspect angle in EOC2, where the training and testing data sets are in mutually exclusive depression angle bins. The system is tested with exemplars from depression angles not trained on. The two data sets are still fairly similar at most aspect angles, with the exception of the contextual values at the extremes.

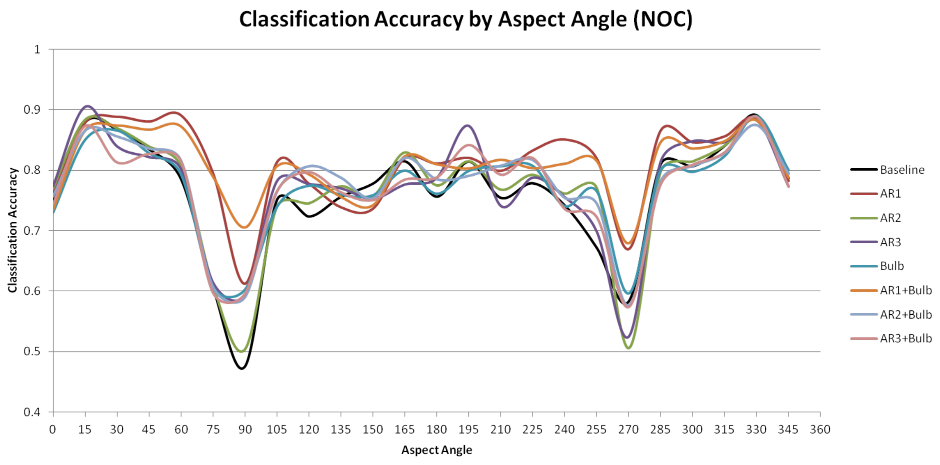


Figure 36. ***Classification Accuracy by Aspect Angle by Contextual Methods in NOC***

The aggregated classifier performance achieved with contextual features were examined at each 15 degree aspect angle, the goal was to find ranges where the different methods complimented each other. By finding these complimenting ranges, the system can chose the optimal contextual features to use given the aspect angle of the object. Figure 36 and 37 shows the template matching classification accuracy by aspect angle for the systems in two different operation conditions. The system performance in NOC is in Figure 36, the classification accuracy remains relatively constant across the different aspect angles, with the exception of the ranges around 90 degrees and 270 degrees. The increase in overall performance from the contextual features are achieved by making up for the shortfalls of the baseline at those particular aspect angles.

The system performance in EOC2 is in Figure 37; the results are much more erratic, with the similar decrease in classification accuracy around 90 degrees and 270 degrees. However, in both cases, there appears to be little complementing behavior in the different methods. The different contextual feature extraction methods seem to follow similar behaviors when aspect angle changes, with different level of effects. Although there is evidence suggesting that AR1 performs better at aspect angles

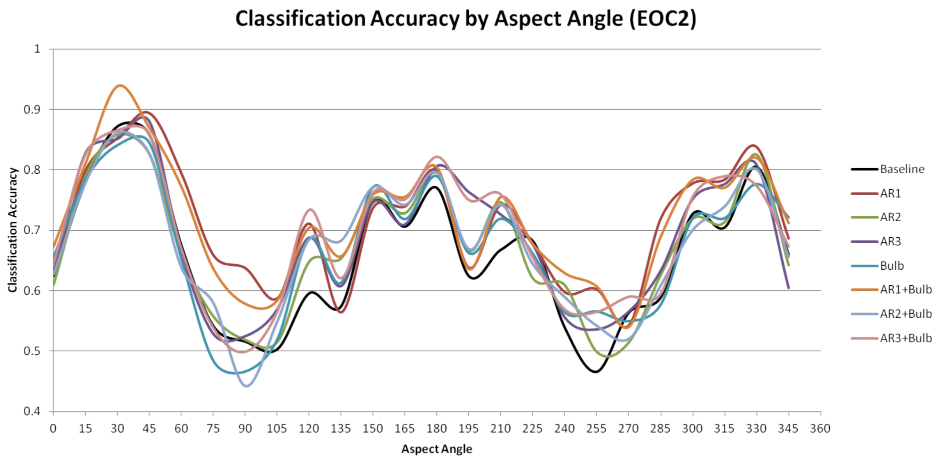


Figure 37. ***Classfication Accuracy by Aspect Angle by Contextual Methods in EOC2***

45-105 degrees, while AR3 performs better at 180-210 degrees. Another interesting evaluation is that AR1 out performances AR1+Bulb at 45-120 degrees and many other locations due to the poor Bulbosity index performance at those aspect angles.

Figure 38 further breaks the contextual data into the 15 different target classes in NOC. Unlike the aggregated values from Figure 34 and 35, the two contextual features data sets are much closer when separated in the individual targets. Note that AR1 shows fairly different charts for target class 1, 2, and 6 in comparison with AR2 and AR3. Figures 40 and 41 show the benefits of using AR1 and Bulbosity by the individual target classes. Notices that the largest increases in CA are from AR1 at class 1, 2, and 6. Figure 39 shows the same class level context data in EOC2. For AR1 and Bulbosity, the test data matches very well against the train data for most the target classes. AR1 and Bulbosity seems to be somewhat invariant to the change in depression angle.

4.2 Improvements on Template Matching

This research examines first the pure effects of contextual features on template matching, without consideration of the OOL or NDEC decisions. One goal of in-

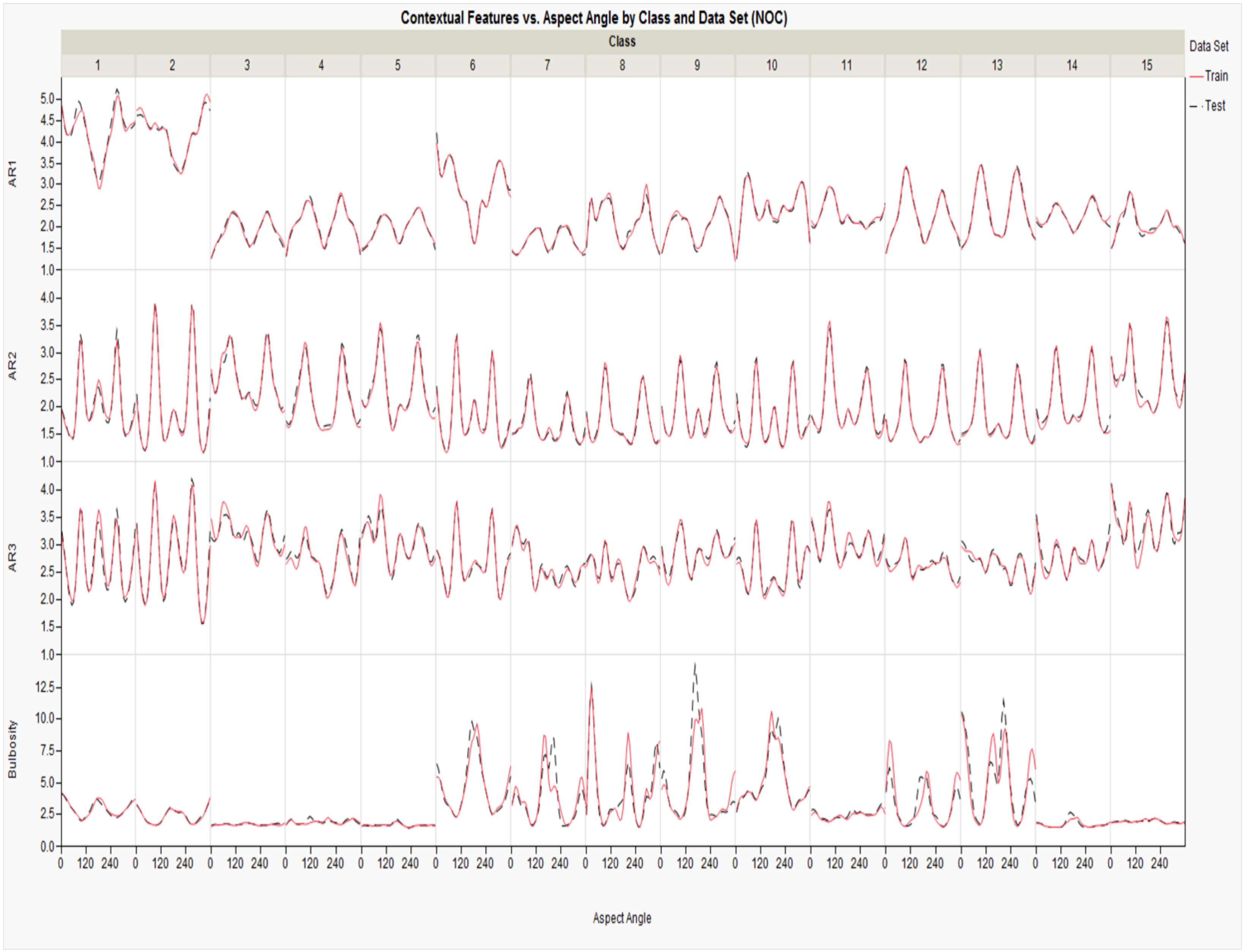


Figure 38. *Contextual Features by Aspect Angle by Target Class by Data Set in NOC*

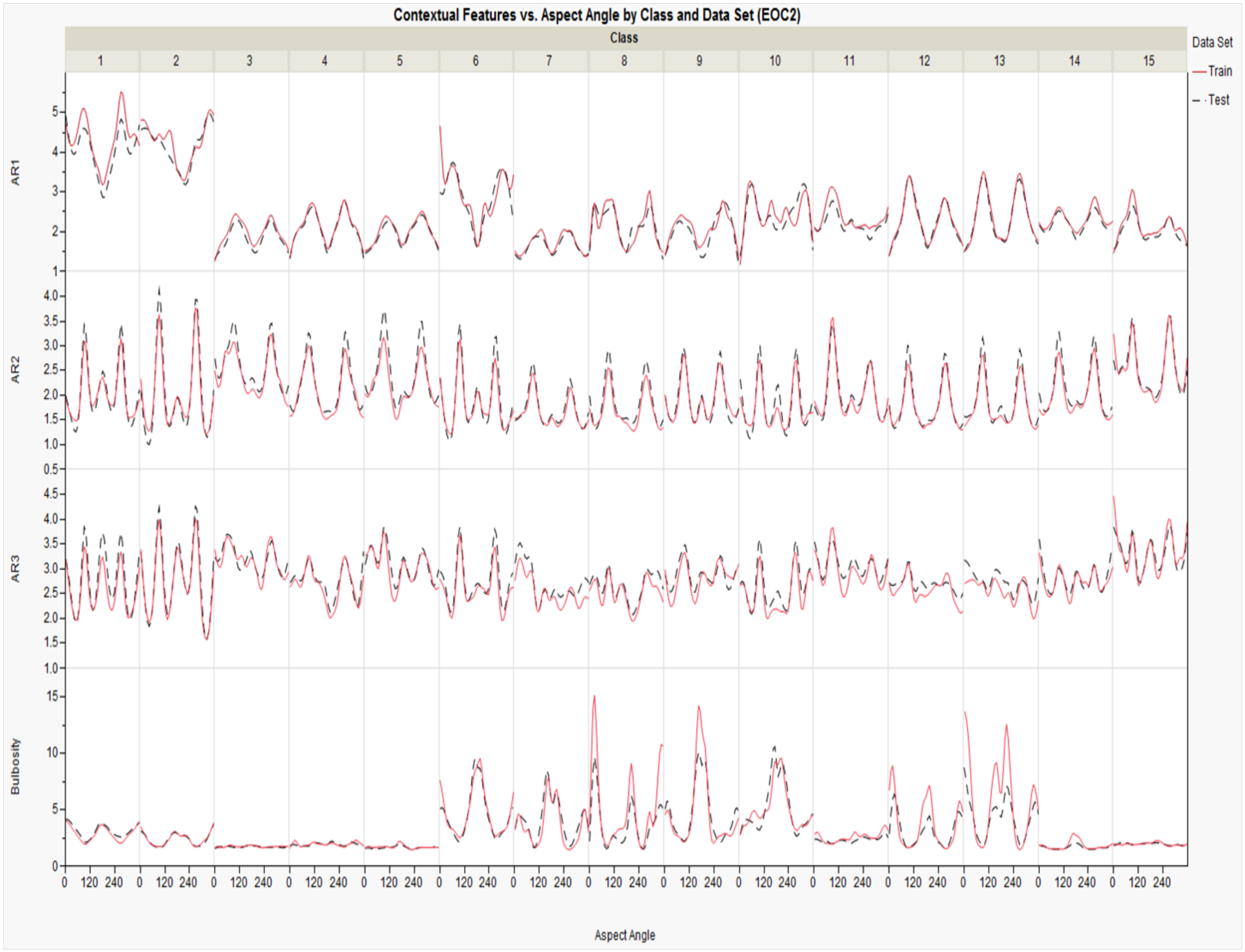


Figure 39. *Contextual Features by Aspect Angle by Target Class by Data Set in EOC2*

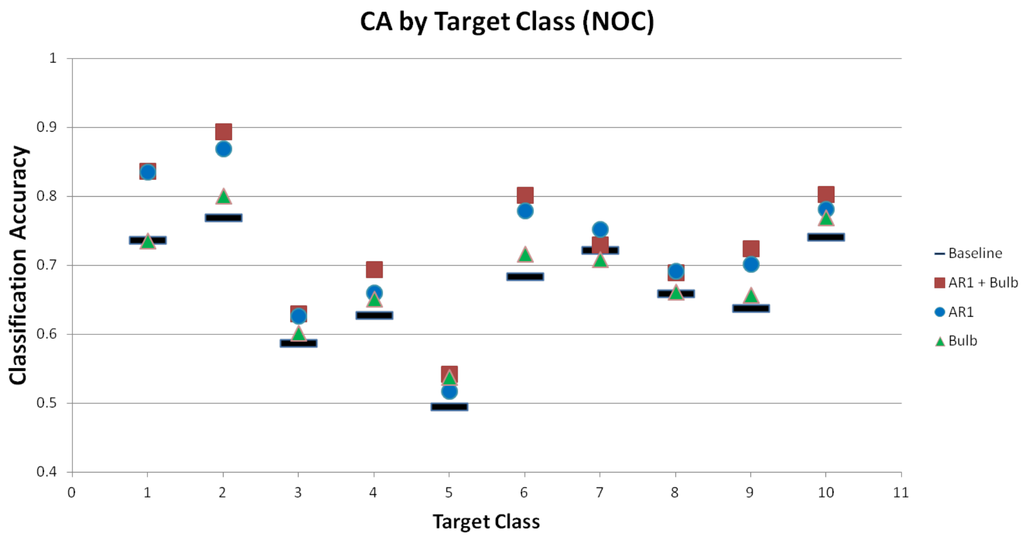


Figure 40. ***CA by Target Class (NOC)***

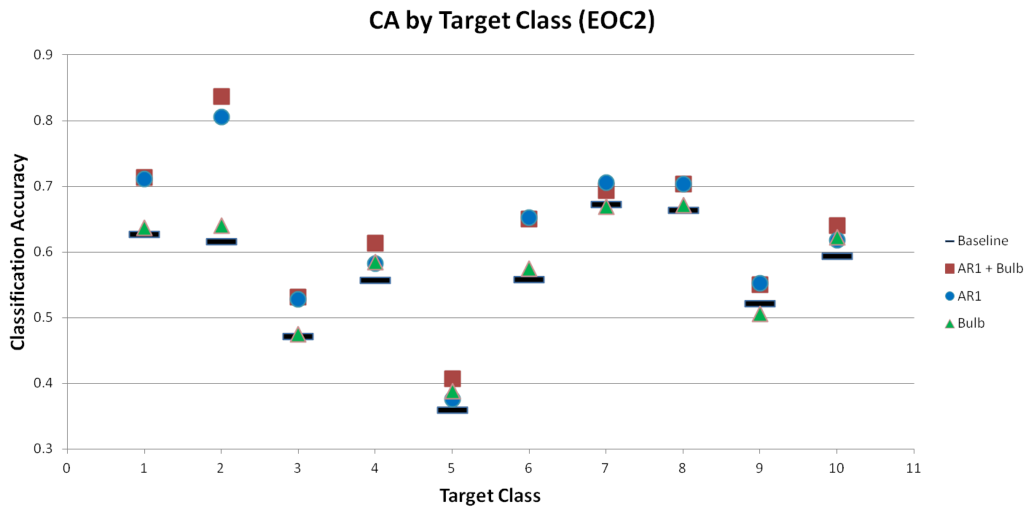


Figure 41. ***CA by Target Class (EOC2)***

cluding additional contextual features is to increase the classification accuracy for the system. The simple template matching setup forces the system to make decisions based solely on the Mahalanobis distances. This avoid the need to vary threshold for OOL and NDEC to have exactly equal levels of performance for comparison. This approach provides an expedient means to compare the baseline classifier performance with classifier performance achieved using additional contextual features. The results in Tables 7, 8, and 9 are system performances in NOC, EOC and EOC2 experiment setups. The NDEC rates are at 100%, all exemplars make declaration. The OOL rates are at 0%, no OOL decisions are made. The only changing variables are the features used in classification. The conditional coloring schemes in the tables indicate the level of improvements over the baseline system, relative to the columns; dark green is higher than the lighter green colors.

Table 7 shows the results from the system at NOC. All seven different treatments with the contextual features improved upon the baseline, which uses only 10 HRR features. AR1 method features performance appears to be the best 11 feature system, with highest improvements over the baseline. The combination of AR1 method, Bulbosity, and HRR features appears to be the best 12 feature system, with the highest improvements over the baseline in every evaluation measure. There is a 7.1% increase in classification accuracy (CA) over the baseline using AR1+bulb in a forced decision mode.

Table 8 shows similar results in EOC, with overall lower classification accuracy; approximately 66% percent of the testing data is not represented in the training set. AR1 method and Bulbosity method is still the best system.

Table 9 shows the system results in EOC2, which is the most challenging scenario, the training data set and test data set consist of data collected at mutually exclusive

Table 7. Force decision classification accuracy in NOC

NOC	CA	CETP	CFTP	Agg CA	Agg ETP	Agg FTP
Baseline	0.663	0.641	0.686	0.881	0.870	0.893
AR1 + Bulb	0.734	0.719	0.749	0.916	0.896	0.936
AR2 + Bulb	0.690	0.670	0.710	0.901	0.883	0.920
AR3 + Bulb	0.694	0.671	0.718	0.900	0.882	0.919
AR1	0.721	0.702	0.741	0.900	0.877	0.923
AR2	0.677	0.657	0.698	0.884	0.870	0.899
AR3	0.681	0.657	0.705	0.885	0.874	0.897
Bulb	0.683	0.665	0.702	0.899	0.884	0.916

Table 8. Force decision classification accuracy in EOC

EOC	CA	CTP	CFP	Agg TCA	Agg ETP	Agg FTP
Baseline	0.617	0.579	0.656	0.853	0.812	0.895
AR1 + Bulb	0.688	0.674	0.702	0.896	0.860	0.933
AR2 + Bulb	0.649	0.621	0.677	0.878	0.838	0.920
AR3 + Bulb	0.658	0.629	0.689	0.881	0.846	0.916
AR1	0.672	0.649	0.695	0.880	0.838	0.922
AR2	0.630	0.596	0.665	0.857	0.814	0.901
AR3	0.638	0.606	0.671	0.859	0.824	0.894
Bulb	0.636	0.603	0.670	0.875	0.833	0.918

depression angles. The AR1 method is still the best single contextual feature and the AR1+Bulbosity is again the best combination.

4.3 Sensitivity Analysis

The previous section's results indicated that the contextual features improve on template matching in a forced decision methodology. The next step in the analysis is to reevaluate classifier performance with the NDEC and OOL classifiers to find the optimal threshold for performance. Figures 42 and 43 shows the classification accuracy results from the systems at NOC and EOC2 by varying the NDEC classifier thresholds. The NDEC threshold ranged from 0 for 100% declaration rate, and 1 for 0% declaration rate. The OOL threshold was held constant at 1, producing relatively

Table 9. Force decision classification accuracy in EOC2

EOC2	CA	CETP	CFTP	Agg CA	Agg ETP	Agg FTP
Baseline	0.563	0.526	0.600	0.825	0.760	0.891
AR1 + Bulb	0.633	0.620	0.646	0.863	0.804	0.922
AR2 + Bulb	0.589	0.554	0.625	0.844	0.777	0.912
AR3 + Bulb	0.601	0.560	0.642	0.846	0.788	0.904
AR1	0.622	0.600	0.645	0.846	0.777	0.916
AR2	0.571	0.529	0.614	0.821	0.749	0.894
AR3	0.583	0.540	0.626	0.823	0.762	0.884
Bulb	0.576	0.544	0.607	0.846	0.784	0.909

constant OOL rates, from 35%-38%. Results near 0% declaration rates are unreliable, a small amount of exemplars are evaluated using the system, and therefore, the results are irrelevant. It would also be unrealistic in real world situation to make such low declaration rates for a classification system.

Figure 42 shows the system's classification accuracy when varying declaration rate at NOC. The optimal rate of declaration appears to be 50% to 60%. As it was discussed in section 2, when the declaration is increase, the classification accuracies decrease. The system's poorest performances are at 100% declaration rate. AR1 appears to be best single feature when added to the baseline. AR1 and Bulbosity are also the best combination of features added to baseline. From 50% to 100% declaration rate, AR1+Bulb averaged 5.0% increased classification accuracy over the baseline system.

Figure 43 shows the system's classification accuracy when varying declaration rate at EOC2. The results are similar to the system at NOC. AR1 and Bulbosity is still the best combination of features added to baseline. From 50% to 100% declaration rate, AR1+Bulb averaged 5.3% increased classification accuracy over the baseline system. The improvement in classification accuracy appears to greater in EOC, even through train and test data sets are in different depression angle ranges. These results support

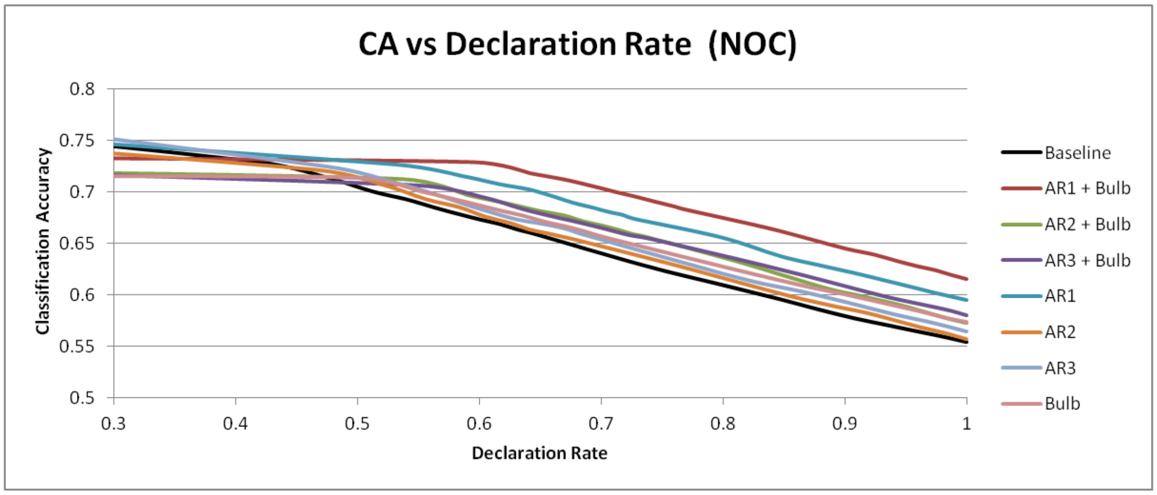


Figure 42. ***Classification Accuracy vs Declaration Rate in NOC***

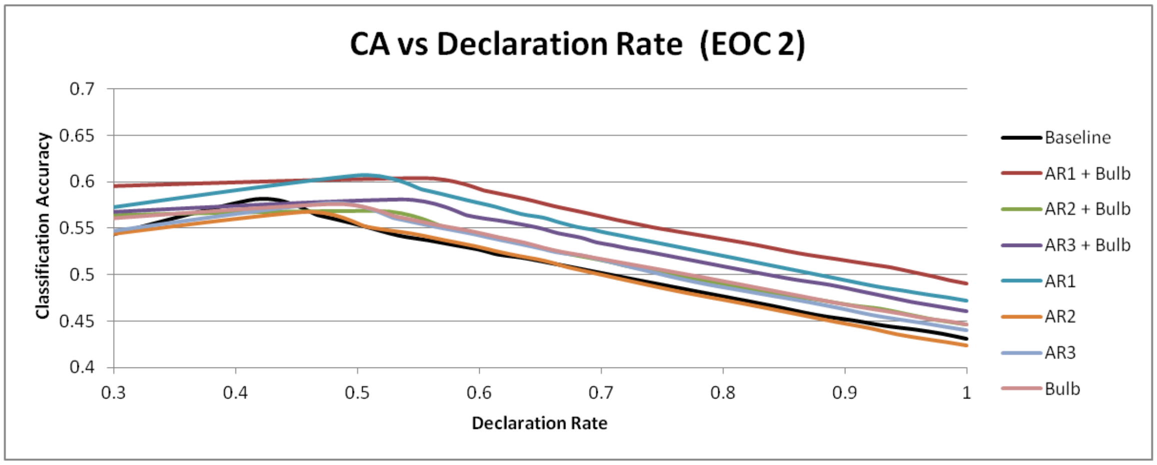


Figure 43. ***Classification Accuracy vs Declaration Rate in EOC2***

the idea that AR1 and the Bulbosity are invariant to the change in depression angle.

The systems were also evaluated by varying NDEC and OOL thresholds. The NDEC threshold ranged from 0 to 1, producing 0% to 100% declaration rate. The OOL threshold ranged from -1, producing 0% OOL rate, to 100, producing near 100%. Since there are few exemplars with extreme large Mahalanobis distances, 100 standard deviations seemed a reasonable for an upper bound threshold for this experiment's purpose. Table 10 shows the system classification accuracies with AR1 and Bulbosity index in NOC. AR1 and Bulbosity appears to be the best combination of features added to the baseline in terms of classification accuracy, only AR1+Bulb are shown. The optimal system performance appears to be from 50%-60% declaration rate and 30-40% OOL rate.

Table 10. AR1+Bulbosity Classification Accuracy with varying Declaration Rates and OOL Rates in NOC

NOC	OOL Rate								
		0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
	0.3				0.813	0.786	0.757	0.727	0.697
	0.4		0.844	0.813	0.778	0.753	0.726	0.697	0.667
	0.5	0.846	0.810	0.773	0.736	0.711	0.685	0.657	0.628
DEC Rate	0.6	0.800	0.758	0.716	0.676	0.650	0.628	0.604	0.577
	0.7	0.744	0.696	0.648	0.601	0.578	0.559	0.539	0.515
	0.8	0.677	0.624	0.571	0.520	0.499	0.483	0.466	0.445
	0.9	0.567	0.508	0.449	0.393	0.376	0.366	0.354	0.340
	1	0.394	0.339	0.284	0.232	0.224	0.219	0.215	0.207

Table 11 shows the system classification accuracies with AR1 and Bulbosity index feature improvements over the baseline in NOC. The system appears to improve upon the baseline in area considered in NOC.

Table 11. AR1+Bulbosity Classification Accuracy Improvements over baseline with varying Declaration Rates and OOL Rates in NOC

NOC	OOL Rate								
		0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
	0.3				0.070	0.076	0.082	0.083	0.078
	0.4		0.063	0.075	0.079	0.085	0.092	0.092	0.087
	0.5	0.051	0.089	0.093	0.094	0.098	0.103	0.103	0.096
DEC Rate	0.6	0.071	0.104	0.103	0.096	0.097	0.103	0.106	0.098
	0.7	0.093	0.120	0.111	0.095	0.094	0.101	0.106	0.100
	0.8	0.095	0.137	0.124	0.103	0.100	0.106	0.109	0.102
	0.9	0.094	0.138	0.112	0.078	0.077	0.083	0.087	0.084
	1	0.082	0.131	0.105	0.065	0.063	0.066	0.070	0.066

Tables 12 and 13 shows the system evaluated at varying NDEC and OOL thresholds in EOC2. Table 12 shows the system classification accuracies with AR1 and

Bulbosity index in EOC2. The optimal system performance appears to be at 50%-60% declaration rate and 30-40% OOL rate. Table 13 also shows that the system improves upon the baseline in area considered in EOC2.

Table 12. AR1+Bulbosity Classification Accuracy with varying Declaration Rates and OOL Rates in EOC2

EOC2		OOL Rate							
		0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
	0.3			0.73348	0.70226	0.66587	0.63741	0.60973	0.58242
	0.4	0.76258	0.7275	0.68983	0.65478	0.61906	0.5935	0.56845	0.54367
	0.5	0.71618	0.67742	0.63866	0.60251	0.56986	0.54564	0.52365	0.4991
DEC Rate	0.6	0.6633	0.6225	0.58169	0.54533	0.5135	0.49045	0.47114	0.44773
	0.7	0.57868	0.54374	0.5088	0.47678	0.44629	0.42546	0.40945	0.38913
	0.8	0.49582	0.46087	0.42593	0.39726	0.37056	0.35293	0.33915	0.32336
	0.9	0.42281	0.37649	0.33017	0.29575	0.27235	0.25979	0.25109	0.24054
	1	0.30251	0.2595	0.21451	0.18088	0.16172	0.15671	0.1527	0.14732

Table 13. AR1+Bulbosity Classification Accuracy Improvements over baseline with varying Declaration Rates and OOL Rates in EOC2

EOC2		OOL Rate							
		0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
	0.3			0.100	0.103	0.097	0.096	0.093	0.090
	0.4		0.096	0.102	0.101	0.093	0.095	0.094	0.090
	0.5	0.059	0.105	0.109	0.105	0.098	0.099	0.099	0.093
DEC Rate	0.6	0.062	0.119	0.116	0.110	0.101	0.101	0.102	0.095
	0.7	0.050	0.120	0.117	0.110	0.100	0.100	0.101	0.094
	0.8	0.049	0.123	0.116	0.108	0.097	0.097	0.098	0.092
	0.9	0.068	0.130	0.108	0.089	0.076	0.077	0.080	0.077
	1	0.068	0.122	0.094	0.070	0.058	0.060	0.062	0.060

4.4 Mahalanobis distance distributions

Since the Mahalanobis distance is a vital characteristic created by the classification systems, their distribution were evaluated at the different conditions. The OOL algorithm utilizes the training set of Mahalanobis distance mean and standard deviation to make OOL decision, the test distribution should match in order to produce high levels classification accuracy. Table 14 shows the mean and standard deviation of the in library target mahalanobis distances at EOC2 for the baseline and different contextual features. The difference in the train mean and test mean divided by the train standard deviation is a metric used to judge on the distributions compare in terms of system performance. The difference in means remain relatively constant. However, the differences relative to training standard deviations are significantly different.

Table 14. Distribution of Mahalanobis Distances in EOC2

Mahal Dist in EOC2	Train Mean	Train Std dev	Test Mean	Test Std dev	Difference/ Train Std dev
Baseline	12.7	9.1	22.7	26.9	1.09
AR1 + Bulb	19.7	14.6	34.6	129.5	1.02
AR2 + Bulb	19.0	14.2	34.7	129.1	1.11
AR3 + Bulb	19.5	13.9	34.6	128.5	1.09
AR1	16.5	11.2	27.4	29.5	0.97
AR2	15.7	10.8	27.4	32.0	1.08
AR3	16.3	10.9	27.3	29.5	1.01
Bulb	15.9	12.5	29.9	127.9	1.12

Figure 44 and 45 shows an example of this effect using the baseline system and AR1+Bulb system. In EOC2, the training and test Mahalanobis distances distributions differ. The test distribution shifts to right, since the test exemplars have further Mahalanobis distances from the training templates. Figure 44 shows the standardize Mahalanobis distance in terms of the training mean and standard deviations in the baseline case; the test distribution is shifted to the right.

Figure 45 shows the distribution of standardize Mahalanobis distances in terms of the training mean and standard deviation in system with AR1 and Bulbosity index features. The distributions match is better, which improved classification performance.

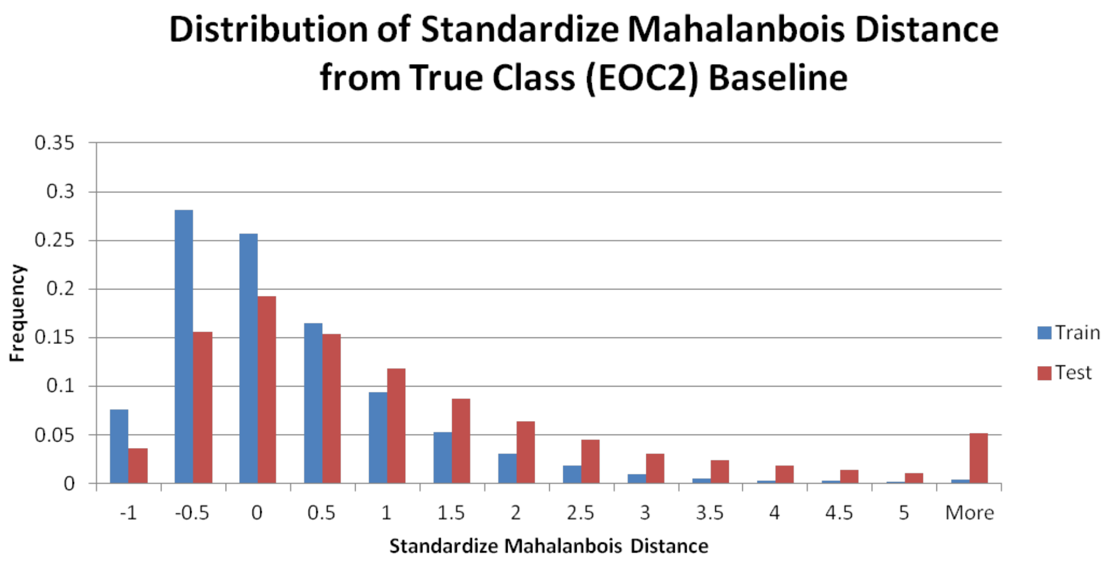


Figure 44. *Baseline Distribution of Standardize Mahalanobis Distances in EOC2*

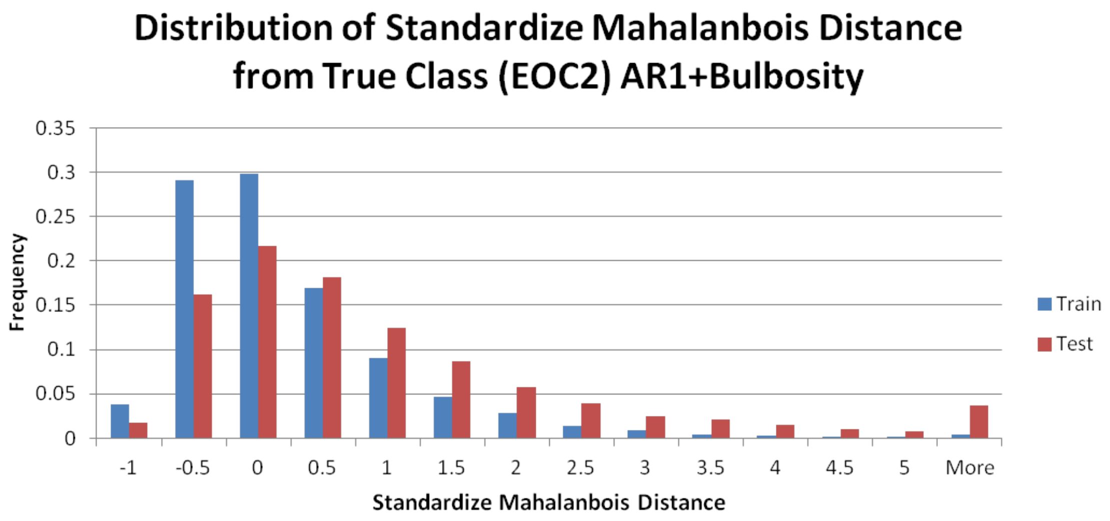


Figure 45. *AR1+Bulbosity Distribution of Standardize Mahalanobis Distances in EOC2*

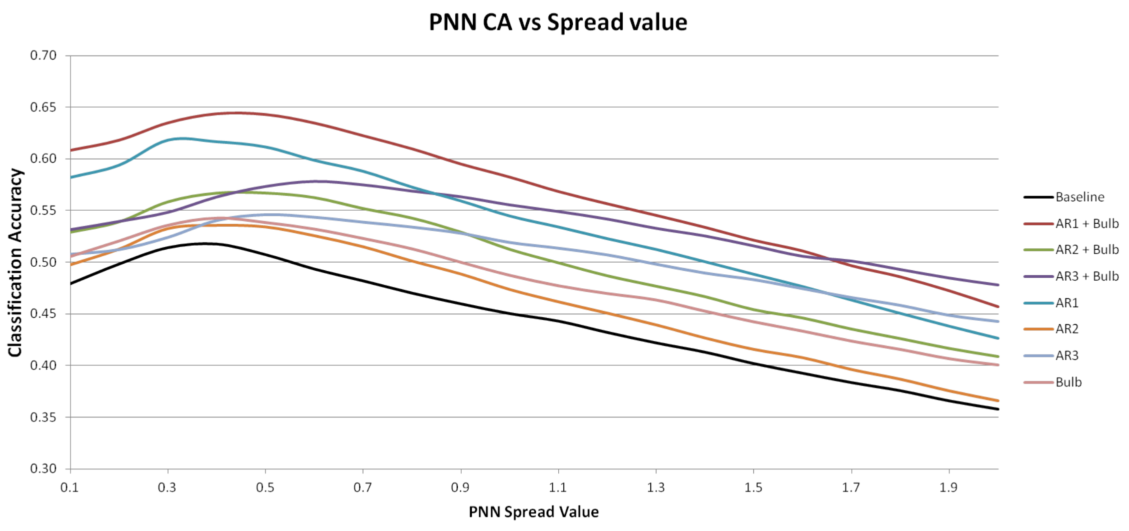


Figure 46. ***Analysis of PNN CA vs Spread value***

4.5 PNN Performance

This research also examines the pure effects of contextual features on PNNs, without consideration of the OOL or NDEC decisions. Similar to the analysis done in Section 4.2, the approach provides an expedient means to compare the baseline classifier performance with classifier performance achieved using additional contextual features. An analysis was also conducted to find the optimal PNN spread value, Figure shows the CA averaged across the conditions as the spread value changes. The data suggest that a spread value of .5 works across all the methods as an optimal point. The results in Tables 15, 16, and 17 are system performances in NOC, EOC and EOC2 experiment setups. There is no consideration for NDEC or OOL in these PNN scenarios. The only changing variables are the features used in classification. The conditional coloring schemes in the tables indicate the level of improvements over the baseline system, relative to the columns; dark green is higher than the lighter green colors.

Table 15 shows the results from the system at NOC. All seven different treatments with the contextual features improved upon the baseline, which uses only 10 HRR features. AR1 method features performance appears to be the best 11 feature system, with highest improvements over the baseline. The combination of AR1 method, Bulbosity, and HRR features appears to be the best 12 feature system, with the highest improvements over the baseline in every evaluation measure. There is a 9% increase in classification accuracy (CA) over the baseline using AR1+bulb in a forced decision mode.

Table 15. PNN Force decision classification accuracy in NOC

NOC	CA	ETP	FTP	Agg CA	Agg ETP	Agg FTP
Baseline	0.598	0.614	0.581	0.843	0.875	0.810
AR1 + Bulb	0.688	0.732	0.643	0.895	0.936	0.852
AR2 + Bulb	0.651	0.682	0.619	0.885	0.922	0.847
AR3 + Bulb	0.661	0.692	0.629	0.873	0.915	0.831
AR1	0.668	0.703	0.631	0.871	0.909	0.832
AR2	0.626	0.648	0.604	0.858	0.888	0.828
AR3	0.644	0.658	0.630	0.845	0.874	0.815
Bulb	0.626	0.657	0.593	0.876	0.919	0.831

Table 16 shows similar results in EOC, with overall lower classification accuracy; approximately 66% percent of the testing data is not represented in the training set. AR1 method and Bulbosity method is still the best system. There is a 9.7% increase in classification accuracy (CA) over the baseline using AR1+bulb in a forced decision mode.

Table 17 shows the system results in EOC2, which is the most challenging scenario, the training data set and test data set consist of data collected at mutually exclusive depression angles. The AR1 method is still the best single contextual feature and the AR1+Bulbosity is again the best combination. The increase in CA is highest in EOC2 using AR1+bulb, there is a 10.2% increase in CA. The additional contextual features improve upon the baseline much more in EOC conditions for PNN.

Table 16. PNN Force decision classification accuracy in EOC

EOC	CA	ETP	FTP	Agg CA	Agg ETP	Agg FTP
Baseline	0.521	0.529	0.512	0.820	0.802	0.838
AR1 + Bulb	0.618	0.675	0.561	0.876	0.884	0.868
AR2 + Bulb	0.568	0.621	0.516	0.858	0.872	0.843
AR3 + Bulb	0.622	0.644	0.600	0.860	0.871	0.849
AR1	0.592	0.618	0.566	0.847	0.835	0.859
AR2	0.546	0.567	0.526	0.830	0.823	0.837
AR3	0.594	0.590	0.597	0.817	0.808	0.827
Bulb	0.555	0.597	0.513	0.856	0.864	0.848

Table 17. PNN Force decision classification accuracy in EOC2

EOC2	CA	ETP	FTP	Agg CA	Agg ETP	Agg FTP
Baseline	0.496	0.503	0.488	0.790	0.764	0.817
AR1 + Bulb	0.597	0.662	0.532	0.862	0.872	0.851
AR2 + Bulb	0.557	0.606	0.508	0.849	0.876	0.822
AR3 + Bulb	0.573	0.614	0.532	0.837	0.857	0.816
AR1	0.566	0.603	0.529	0.818	0.807	0.829
AR2	0.520	0.544	0.496	0.808	0.808	0.808
AR3	0.547	0.551	0.543	0.792	0.780	0.804
Bulb	0.537	0.575	0.499	0.839	0.845	0.834

4.6 BNT Performance

The results section 4.2 established that higher classification accuracy is achieved when the systems are at 50% declaration rate than at 100% declaration. The research used these two thresholds as the upper bound and lower bound of the systems' performance used to evaluate the performance of the Bayesian Belief Networks. The BNT frameworks used in this research produce results only at 100% declaration rates, there are no NDEC decisions made on BNT posterior probabilities. Therefore, the BNT classification accuracies should be at minimum equal to the system at 100% declaration without BNT.

Two different Bayesian networks were evaluated in this research; the first is with OOL, NDEC and template matching nodes with their corresponding two polarities, described in Figure 31. This network has no contextual information nodes, and is referred to as BNT no Context Node in the study. The second is the full BNT network illustrated in Figure 32; this network includes the additional contextual information nodes. The full BNT network takes advantage of the relatively invariant characteristic of the contextual features between the training and testing data sets.

Figure 47 shows the classification accuracy of the BNT method using the different contextual features along with the baseline in NOC. All of the different feature sets' performances improve upon the systems with 100% declaration and no BNT. Their classification accuracies fall short of the optimal performance set for each system at 50% declaration, with the exception of feature set with AR1 and Bulbosity index. The AR1+Bulb feature set added to baseline HRR features improves beyond the optimal performance with no BNT at 50% declaration, while producing results at 100% declaration rate using BNT. This has significant impact to the warfighter. The BNT system provides the warfighter twice as many declarations with a 5.4% improved accuracy over the baseline model's best performance. The Full BNT method's improvements, shown in Figure 48, over system with no BNT are much more apparent in EOC2, particular by AR1 alone and AR1+Bulb. The BNT system provides the warfighter twice as many declarations with a 7.1% improved accuracy over the baseline model's best performance. Tables 18 and 19 shows the detailed results of the BNT analysis with the CA and OOL rates.

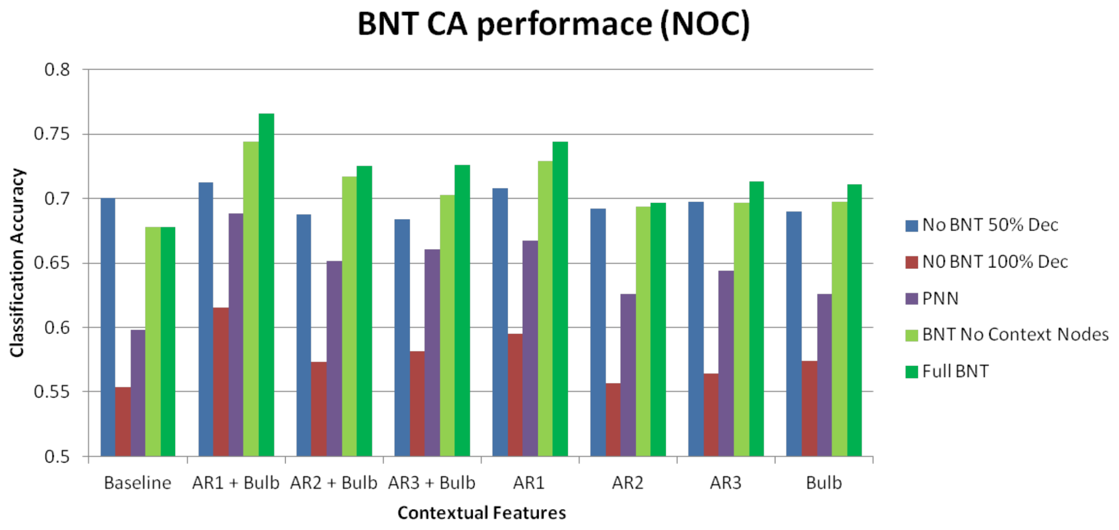


Figure 47. ***BNT Classfication Accuracy in NOC***

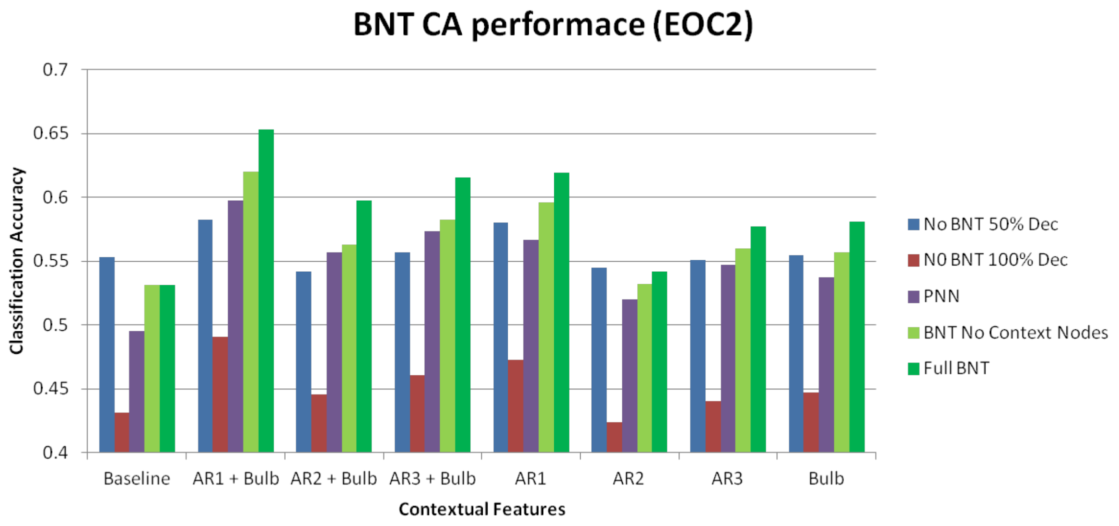


Figure 48. ***BNT Classfication Accuracy in EOC2***

Table 18. BNT CA and OOL detailed results in NOC

NOC	No BNT 50% Dec		No BNT 100% Dec		Full BNT	
	CA	OOL	CA	OOL	CA	OOL
Baseline	0.700	0.359	0.554	0.358	0.678	0.413
AR1 + Bulb	0.712	0.415	0.616	0.431	0.766	0.444
AR2 + Bulb	0.688	0.382	0.573	0.379	0.725	0.392
AR3 + Bulb	0.684	0.375	0.582	0.377	0.726	0.470
AR1	0.708	0.413	0.595	0.429	0.744	0.435
AR2	0.692	0.371	0.557	0.374	0.697	0.411
AR3	0.697	0.366	0.564	0.367	0.713	0.494
Bulb	0.690	0.375	0.574	0.363	0.711	0.385

Table 19. BNT CA and OOL detailed results in EOC2

EOC2	No BNT 50% Dec		No BNT 100% Dec		Full BNT	
	CA	OOL	CA	OOL	CA	OOL
Baseline	0.553	0.360	0.431	0.352	0.532	0.476
AR1 + Bulb	0.582	0.415	0.491	0.418	0.653	0.459
AR2 + Bulb	0.542	0.379	0.446	0.388	0.598	0.440
AR3 + Bulb	0.557	0.349	0.460	0.362	0.616	0.468
AR1	0.580	0.423	0.473	0.431	0.619	0.474
AR2	0.545	0.391	0.424	0.392	0.542	0.459
AR3	0.551	0.368	0.440	0.366	0.577	0.469
Bulb	0.554	0.324	0.447	0.327	0.581	0.440

V. Summary and Conclusion

This research demonstrates that augmenting traditional HRR features with contextual features significantly improves classification accuracy of template classifiers. These benefits are most readily observed when performing classification of targets in extended operating conditions. Additional improvements to classification accuracy are achieved with the fusion of multiple classification methods and contextual information using Bayesian Belief Networks.

5.1 Research Contributions

1. Developed and analyzed the effectiveness of 4 methods to extracting contextual features from SAR images.
2. Showed the benefits of using contextual features in addition to HRR features in classification of SAR ground targets.
3. Analyzed the gain/loss relationship between OOL and NDEC method by adjusting thresholds and identified the optimal OOL and NDEC thresholds for performance of the classification system.
4. Created the process and demonstrate the potential benefits of using a Bayesian Belief Network to fuse feature information and decisions from OOL, NDEC, template matching, and Probabilistic Neural Network classifiers from multiple sensors.

5.2 Limitations

1. The transformation method shown for segmentation assumes that the SAR target objects are at the center of the image. In real world application, the center of the object must be located prior to transformation of the image intensities.

2. The data set used in this research contains relatively a small set of targets for classification, performance in a real world system with hundreds of target classes and OOL target classes may degrade.

5.3 Future Research

1. Due to the high computational demands of extracting features from the entire data set of images for every reasonable threshold, sensitivity analysis was not performed on the threshold for segmentation. Further research is required to find the optimal threshold for segmentation process.

2. Identify process for extracting additional independent contextual features, such as object height from images.

3. Study the effects of adding dependant relationships between the BNT nodes.

Appendix A

Quad-Chart

Combat Identification of Synthetic Aperture Radar Images using Contextual Features and Bayesian Belief Networks

Capt John Situ

Advisor: Dr. Mark A. Friend   Reader: Dr. Kenneth W. Bauer

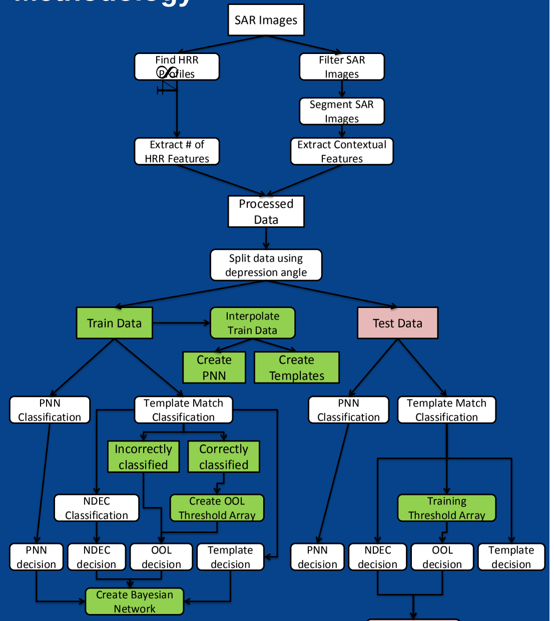
Department of Operational Sciences (ENS)

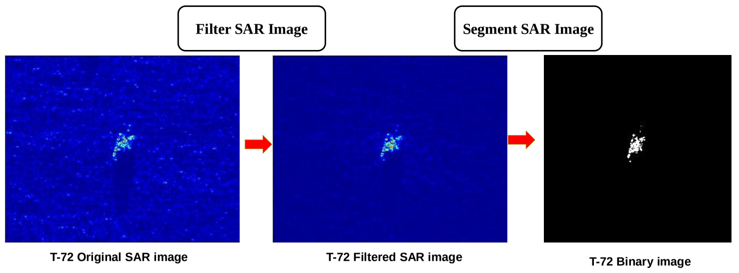
Air Force Institute of Technology

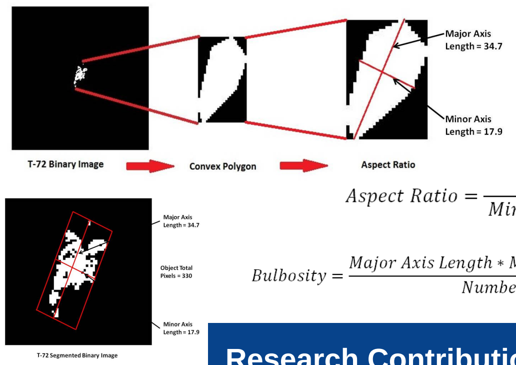
Research Objectives:

- Find contextual features from SAR images
- Use contextual features in conjunction with HRR features to improve classification accuracy at Normal or Extended Operating Conditions.
- Improve classification accuracy by fusing multiple classifier outputs using Bayesian Belief Networks and the direct use of the contextual features.

Methodology



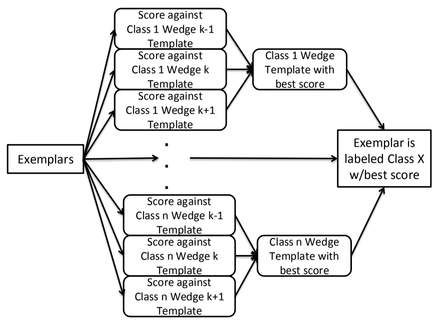




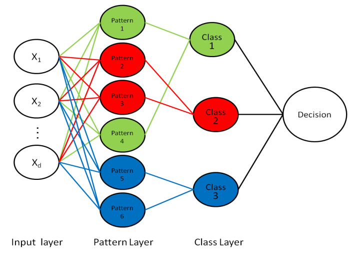
$$Aspect\ Ratio = \frac{Major\ Axis\ Length}{Minor\ Axis\ Length\ Length}$$

$$Bulbosity = \frac{Major\ Axis\ Length * Minor\ Axis\ Length\ Length}{Number\ of\ pixels}$$

Template Matching Classification


Exemplars
Score against Class 1 Wedge k-1 Template
Score against Class 1 Wedge k Template
Score against Class 1 Wedge k+1 Template
Score against Class n Wedge k-1 Template
Score against Class n Wedge k Template
Score against Class n Wedge k+1 Template
Class 1 Wedge Template with best score
Class n Wedge Template with best score
Exemplar is labeled Class X w/best score

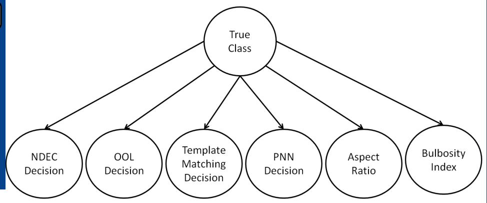
Probabilistic Neural Network (PNN)



Research Contributions:

- Developed 4 methods of contextual feature extraction
- Demonstrated the utility of Bayesian Belief Network to fuse multiple classifiers outputs and multiple sensors
- Method using contextual features provides the warfighter with twice the number of decisions and at a higher accuracy level

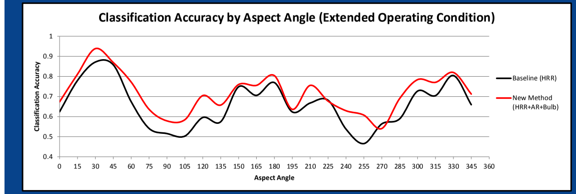
Bayesian Belief Network (BNT)

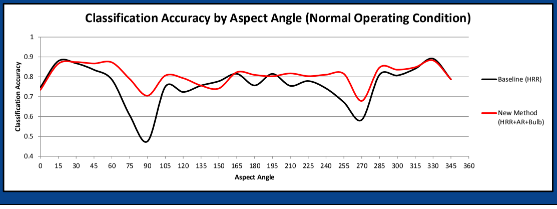


$$P(Class_i|NDEC,OOL,TM,PNN,AR,Bulb) = \frac{P(NDEC,OOL,TM,PNN,AR,Bulb|Class_i)}{\sum_{j=1}^{n}[P(NDEC|Class_j)*P(OOL|Class_j)*P(TM|Class_j)*P(PNN|Class_j)*P(AR|Class_j)*P(Bulb|Class_j)]}$$

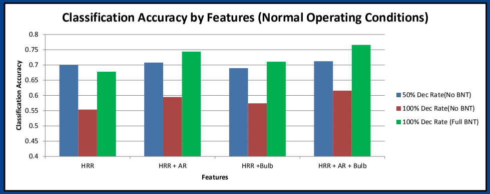
Results:

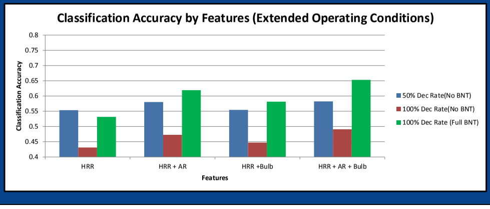
Overall improvements in classification accuracy using contextual features





Further improvements using Bayesian Belief Network





Appendix B
MATLAB code for Segmentation

MATLAB Code for Segmentation of contextual features

```
%================================================
%Data_sort.m file
%This script gathers all the .mat files its current folder and organize the
%target looks by actual depression angles. It will organize the target .mat
%files and call get_context to extract contextual features.
%====================================================
 clear all

tStart = tic; %start timer;

files = dir('*.mat'); %create directory of .mat files from folder

%filelist =zeros(length(files),1);

for i = 1: length(files) %Extract target number/class from file name and add
field title "targetnumber" to file directory
    tempspace = findstr('_',files(i).name);
    files(i).targetnumber =
str2double(files(i).name(tempspace(2)+1:tempspace(3)-1));
    %filelist(i,1) = str2double(files(i).name(tempspace(2)+1:tempspace(3)-
1));
end
%The list of target numbers/class for training
%targetlist = [1, 2, 5, 6, 7, 10, 11, 12, 13, 15];
targetlist = 1:15;
%Index of actually ordered class numbers for F/E/O charts
orderednumbers=[1,1; 2,2; 3,11; 4,12; 5,3; 6,6; 7,7; 8,13; 9,14; 10,4; 11,8;
12,9; 13,5; 14,15; 15,10;];

for z = targetlist %Loop for each target number/class
    class_start = tic;
    %create separated struct for each target class and by polarization
    eval(['target_' num2str(z) '_VV = struct([]);']);
    eval(['target_' num2str(z) '_HH = struct([]);']);

    %for j = 1:10 %Loop through selcted number of .mat files in folder
    for j = 1:length(files) %Loop through all .mat files in folder
        if files(j).targetnumber == z %Check if the current .mat file match
current target number
            eval(['load ' files(j).name]); %Load the current .mat file
            for k = 1: length(Target) %Loop thought every target structure in
current .mat file

                if length(Target(k).hrrProfile2) ~= 322 %check number hrr
profile data points
                     continue; %exit current for loop and don't use datapoint
                end
                 %check the chip matrix size, must be 256 by 256
                if or(size(Target(k).chip,1) ~= 256,size(Target(k).chip,2) ~=
256)
                     continue; %exit current for loop and don't use datapoint
                end
                    if strcmp(Target(k).Polarization, 'HH') %Sepearate by
polarity
```

```
                        tempstruct1 = struct;
                        %assign fields of interest to the sorted struct
                        tempstruct1(1).type = Target(k).TargetType;
                        tempstruct1(1).number = z;
                        tempstruct1(1).order_number = orderednumbers(z,2);
                        tempstruct1(1).filename = files(j).name;
                        tempstruct1(1).structnumber = k;
                        tempstruct1(1).polar = Target(k).Polarization;
                        tempstruct1(1).aspect = Target(k).AspectAngle;
                        tempstruct1(1).depress = Target(k).Depression;
                        tempstruct1(1).d_depress =
Target(k).Desired_Deprssion;
                        tempstruct1(1).hrr = Target(k).hrrProfile2;
                        %get contextual info
                        %tempstruct1(1).chip = Target(k).chip;
                        [aspect_ratio1, aspect_ratio2,aspect_ratio3, bull] =
get_context(Target(k).chip);
                        if aspect_ratio1 == 0;
                            continue;
                        end
                        tempstruct1(1).aspect_ratio1 =aspect_ratio1;
                        tempstruct1(1).aspect_ratio2 =aspect_ratio2;
                        tempstruct1(1).aspect_ratio3 =aspect_ratio3;
                        tempstruct1(1).bulbosity = bull;
                        %add data to the class and polarity struct
                        eval(['target_' num2str(z) '_HH = [' 'target_'
num2str(z) '_HH, tempstruct1(1)];']);

                    elseif strcmp(Target(k).Polarization, 'VV') %Separate by
polarity
                        tempstruct2 = struct;
                        %assign fields of interest to the sorted struct
                        tempstruct2(1).type = Target(k).TargetType;
                        tempstruct2(1).number = z;
                        tempstruct2(1).order_number = orderednumbers(z,2);
                        tempstruct2(1).filename = files(j).name;
                        tempstruct2(1).structnumber = k;
                        tempstruct2(1).polar = Target(k).Polarization;
                        tempstruct2(1).aspect = Target(k).AspectAngle;
                        tempstruct2(1).depress = Target(k).Depression;
                        tempstruct2(1).d_depress =
Target(k).Desired_Deprssion;
                        tempstruct2(1).hrr = Target(k).hrrProfile2;
                        %get contextual info
                        %tempstruct2(1).chip = Target(k).chip;
                        [aspect_ratio1, aspect_ratio2,aspect_ratio3, bull] =
get_context(Target(k).chip);
                        if aspect_ratio1 == 0;
                            continue;
                        end
                        tempstruct2(1).aspect_ratio1 =aspect_ratio1;
                        tempstruct2(1).aspect_ratio2 =aspect_ratio2;
                        tempstruct2(1).aspect_ratio3 =aspect_ratio3;
                        tempstruct2(1).bulbosity = bull;
                        %add data to the class and polarity struct
                        eval(['target_' num2str(z) '_VV = [' 'target_'
num2str(z) '_VV, tempstruct2(1)];']);
```

```
                    end %end if for Polarization compare
                end %end loop for target looks .mat file
            end %end if for matching .mat file to target number/class
        end %end for all .mat files in folder
    classtime = toc(class_start);
    disp(['Target class ', num2str(z) ' done in ', num2str(classtime), '
secs']);
end%end loop for current target number/class
donetime = toc(tStart);
disp(['Training Data Collection Complete in ', num2str(donetime) ' secs']);

clearvars -except target_* %clear all variables excpet ones that start with
target_
save([pwd,'\Sorted\alldata']);

beep;


%=========================================================================
%function get_context
%This function will extract contexutal features from a SAR chip image
%Input: m x n image matrix
%Output:AR1, AR2, AR3, Bulbosity Index
%=========================================================================
function [aspect_ratio1,aspect_ratio2,aspect_ratio3,
bull1,info]=get_context(chip)

[m, n]=size(chip);

%Transformation of Image, sqrt of dist from the center of image
adjusted = zeros(m,n);
for i = 1:m
    for j = 1:n
        dist1 = abs(i-m/2);
        dist2 = abs(j-n/2);
        dist = max(sqrt(dist1^2+dist2^2),.5);
        adjusted(i,j) = chip(i,j)/sqrt(dist);
    end
end

%Find the stats of the image
mu=mean2(adjusted);
sigma = std2(adjusted);
%Define the thresholds
threshold1 = 3*sigma;
threshold2 = 6;

%Blank Binary Image
filtered_image = zeros(m,n);

%Fill in the binary image according to intensity and Thresholds
for i = 1:m
    for j = 1:n
        if adjusted(i,j)> (mu+threshold1)
            %Calls adjacentcells function to evaluate the 8 adjacent cells
            pixels = adjacentcells(adjusted,i,j,threshold1);
            if pixels >= threshold2
                %target pixel
                filtered_image(i,j) = 1;
            end
        else %background
            filtered_image(i,j) = 0;
        end
    end
end

%Use Regionprops to calculate the Major and Minor Axis of binary image
if max(max(filtered_image)) > 0
    region_info = regionprops(filtered_image,'all');
    %Find AR1
    aspect_ratio1 = region_info.MajorAxisLength/region_info.MinorAxisLength;
    if aspect_ratio1 > 10
        aspect_ratio1 = 10;
    end
    %Find Bulbosity Index
    bull1 = region_info.ConvexArea/region_info.Area;
    info = region_info;
else
    aspect_ratio1 = 0;
    bull1 = 0;
    info = 0;
end

%AR2 Method. PCA on orginal Image
[COEFF,SCORE,latent] = princomp(zscore(chip));
%Find AR2
aspect_ratio2 = latent(1)/latent(2);
if aspect_ratio2 > 10
    aspect_ratio2 = 10;
end

%AR3 Method. PCA on Binary Image
[COEFF,SCORE,latent] = princomp(filtered_image);
%Find AR3
aspect_ratio3 = latent(1)/latent(2);
if aspect_ratio3 > 10
    aspect_ratio3 = 10;
end




%=================================================
%Data_Combine.m file
%This script gathers all the .mat files its current folder and combines all
%into one .mat file with each exemplar as a struct. HH Polarity will the
%first array of structures, VV will be the second array.
%=====================================================
clear all;
%start timer
start = tic;
```

```
addpath([pwd,'\Sorted']);

%load sorted data from Data_sort.m
load alldata.mat;

for i = 1:15 %loop for all target class files
    class_start = tic;
    count = 0;
    done = 0;
    %find which polarity the exemplar is in
    eval(['HH = target_' num2str(i) '_HH;']);
    eval(['VV = target_' num2str(i) '_VV;']);
    %organize the files so that the exemplar polarity pairs match eachother
    %in their ordered place in the two array of structures.
    if length(HH) == length(VV)
        done = 1;
        combined = [HH;VV];
        for j = 1: length(HH)
            if or(HH(j).aspect ~=VV(j).aspect, HH(j).depress ~=VV(j).depress)
                disp(['Mismatch at ', num2str(j)]);
                done = 0;
            end
        end
    end

    if done == 0
        for j = 1:length(HH)
            for k = 1:length(VV)
                if  and(HH(j).aspect ==VV(k).aspect, HH(j).depress
==VV(k).depress)
                    count = count + 1;
                    combined(1,count)= HH(j);
                    combined(2,count)= VV(k);
                    continue;
                end
            end
        end
    end

    %store the structures back in alldata.
    eval(['target_' num2str(i) ' = combined;']);
    clear var combined;
    eval(['clear var target_' num2str(i) '_HH;']);
    eval(['clear var target_' num2str(i) '_VV;']);
    class_time = toc(class_start);
    %disp(['Target class ', num2str(i) ' done in ', num2str(class_time), '
secs']);
end
donetime = toc(start);
disp(['Training Data Filter Complete in ', num2str(donetime) ' secs']);
save([pwd,'\Combined\alldata'],'target_*');
```

Appendix C

MATLAB code for Classification

MATLAB code for Classification

```
%==========================================
%Build_Data.m
%Script for organizing all the DCS data post-processsed into
%Training and Testing Sets.

%John Situ
%AFIT/ENS
%last update 18Jan12
%==========================================
tStart = tic;

%establish folder paths
addpath([pwd, '\data']);

%load settings
load([pwd, '\settings.mat']);

%load data file that contain all the data
load([pwd, '\data\alldata.mat']);

%create a list of structures in alldata file
list=whos('-file', [pwd, '\data\alldata']);
numstruct = length(list);

datasplit = settings.datasplit;

%for i = 1:2 %loop through all the class structures
for i = 1:numstruct %loop through all the class structures
    tempstruct = struct;
    eval(['tempstruct = ' list(i).name ';']) %assign current structure to
tempstruct
    tempstruct = tempstruct'; %remove this after fix

    %sort and average the data points, number of observation averaged
    %is set by settings.num_obs.  If =1, then no average actions.
    tempstruct = sort_data(tempstruct);
    if settings.avg_obs ==0
        tempstruct = avg_data(tempstruct);
    end

    %Also need to fix numelements = list(i).size(1);
    numelements = size(tempstruct,1); %find the number of elements in struct
    target_number = tempstruct(1,1).number;
    trainstruct = struct([]);
    teststruct = struct([]);
    count1 =1;
    count2 =1;
    for j = 1:numelements
        depress = tempstruct(j,1).depress;
        if datasplit == 1 %Normal Condidtions - even split. 25% <8 train, 25%
>8 train, 25% <8 test, 25% >8 test.
            if depress <= 8
                count1 = count1 +1;
                if mod(count1,2)==0
```

```
                    trainstruct = [trainstruct; tempstruct(j,:)];
                else
                    teststruct = [teststruct; tempstruct(j,:)];
                end
            elseif depress > 8
                count2 = count2 +1;
                if mod(count2,2)==0
                    trainstruct = [trainstruct; tempstruct(j,:)];
                else
                    teststruct = [teststruct; tempstruct(j,:)];
                end
            end
        elseif datasplit == 2 %<8 half training, half test. >8 test.
            if depress <= 8
                count1 = count1 +1;
                if mod(count1,2)==0
                    trainstruct = [trainstruct; tempstruct(j,:)];

                else
                    teststruct = [teststruct; tempstruct(j,:)];
                end
            elseif depress > 8
                teststruct = [teststruct; tempstruct(j,:)];
            end
        elseif datasplit ==3 %<8 train, >8 test.
            if depress <= 8
                trainstruct = [trainstruct; tempstruct(j,:)];
            elseif depress > 8
                teststruct = [teststruct; tempstruct(j,:)];
            end
        end
    end% End for current data point
    if settings.average_polarity == 1
        %Add an additional set of data from the average of the two polarities
        [trainstruct] = avg_polar(trainstruct);
        [teststruct] = avg_polar(teststruct);
    end
    eval([list(i).name '=trainstruct;'])
    save([pwd,'\data\traindata'],list(i).name,'-append');
    eval([list(i).name '=teststruct;'])
    save([pwd,'\data\testdata'],list(i).name,'-append');
end %end loop for target class

time = toc(tStart);
disp(['Data Split Done in ' num2str(time) ' secs']);




function Build_Template
%================================================================
%Build Templates from training data
%John Situ
%AFIT/ENS
%04Jan12
%================================================================
tStart = tic; %start timer

%establish folder paths
addpath([pwd, '\data']);
addpath([pwd, '\Functions']);
%load settings
load([pwd, '\settings.mat']);

%load training data file
load([pwd, '\data\traindata.mat']);
%create list of structures in training data file
list=whos('-file', [pwd, '\data\traindata']);
numstruct = length(list);

%Define the number features from hrr profiles
numhrrfeatures = settings.num_hrrfeatures;
wedge_degrees = settings.wedge_degrees;

%for i = 1:1 %test run loop
for i = 1:numstruct %loop through all the class structures
    tempstruct = struct;
    eval(['tempstruct = ' list(i).name ';']) %assign current structure to
tempstruct
    targetnumber = tempstruct(1,1).number;
    %if the target class is not in the library, skip loop, no template
    if ismember(targetnumber, settings.outlib_targets)
        continue;
    end

    for j = 1:settings.num_polar %loop for all polarities
        targetclass = tempstruct(1,1).order_number; %assign the current
target class number
        aspectangles = [tempstruct(:,j).aspect]'; %array of aspect angles

        %combine all hrrprofiles for each class&polarity into one matrix
        hrrprofiles = [tempstruct(:,j).hrr]';

        %call interpolation function for hrr data
        %input:(aspect angle, hrr, degree increments)
        interp_hrr = feature_interpolation(aspectangles,hrrprofiles,1);

        %call interpolation function for context data
        %input:(aspect angle, contextdata, degree increments)

context=[tempstruct(:,j).aspect_ratio1;tempstruct(:,j).aspect_ratio2;tempstru
ct(:,j).aspect_ratio3;tempstruct(:,j).bulbosity]';
        interp_context = feature_interpolation(aspectangles,context,1);
        [interp_context]=feature_moving_average(interp_context);

        %call get_hrr_features function
        %input (hrrprofiles, num of features, bin parameter max or mean)
        hrrfeatures =
get_hrr_features(interp_hrr,numhrrfeatures,settings.hrr_maxormean,settings.hr
r_bins);
```

```
        %call build_interp_features function
        %will turn interpolated features into equal size wedges

%[wedge_mu,wedge_sigma]=build_interp_wedges(hrrfeatures,wedge_degrees);

        %combine hrr features and context features.
        %user will define which aspect ratio to use
        %interp_context(:,4) is bulbosity
        if and(settings.aspect_ratio ==0, settings.bulbosity ==0) %no context
            comb_features = hrrfeatures;
        elseif and(settings.aspect_ratio == 0, settings.bulbosity >0) %No
aspect ratio
            comb_features = [hrrfeatures,interp_context(:,4)];
        elseif and(settings.aspect_ratio > 0, settings.bulbosity == 0) %No
bulbosity
            comb_features = [hrrfeatures,...
                interp_context(:,settings.aspect_ratio)];
        else %use both context data
            comb_features = [hrrfeatures,...
                interp_context(:,settings.aspect_ratio),interp_context(:,4)];
        end

        num_wedges = 360/wedge_degrees; %find number of wedges
        for k = 1:num_wedges %loop for each wedge
            if k ==1;%first wedge starts from 1 to wedge_degrees
                start_wedge = 1;
                end_wedge = wedge_degrees;
            else %following wedges plus wedge size to previous end_wedge
                start_wedge = end_wedge + 1;
                end_wedge = end_wedge + wedge_degrees;
            end
            %build temporary matrix of only data with aspect angle in wedge
            temp_wedge = comb_features(start_wedge:end_wedge,:);
            temp_mu = mean(temp_wedge,1);
            %temp_sigma = cov(temp_wedge,1);
            temp_sigma = diag(var(temp_wedge,0,1));

            %store features into template structure ordered by target class
            train_template(targetclass,k,j).wedge_mu=temp_mu;
            train_template(targetclass,k,j).wedge_sigma=temp_sigma;
            %train_template(targetclass,k,j).wedge_sigma2=temp_sigma2;
        end
    end %end loop for polarity
end %end loop for target class
time = toc(tStart);
%Display Done messagge
disp(['Template Done in ' num2str(time) ' secs']);

%save template file
save([pwd,'\data\traintemplate'],'train_template*');


function [train_exemplars]=Build_Train
%================================
%Set up the training data. Establish features for each exemplar.
%John Situ
```

```
%AFIT/ENS
%Last Updated: 04Jan12
%==============================

tStart = tic; %start timer
%establish folder paths
addpath([pwd, '\data']);
addpath([pwd, '\Functions']);
%load settings
load([pwd, '\settings.mat']);

%load training data file
load([pwd, '\data\traindata.mat']);
%create list of structures in training data file
list=whos('-file', [pwd, '\data\traindata']);
numstruct = length(list);

%Define the number features from hrr profiles and wedge sizes
num_hrrfeatures = settings.num_hrrfeatures;

%setup training data structure
train_exemplars = struct([]);
for i = 1:numstruct %loop through all the class structures
    tempstruct = struct;
    eval(['tempstruct = ' list(i).name ';']) %assign current structure to
tempstruct
    for j = 1:settings.num_polar %loop for all polarities
        %combine all hrrprofiles for each class&polarity into one matrix
        hrrprofiles = [tempstruct(:,j).hrr]';

        %call get_hrr_features function
        %input (hrrprofiles, num of features, bin parameter maxormean,
hrrbins)
        hrrfeatures =
get_hrr_features(hrrprofiles,num_hrrfeatures,settings.hrr_maxormean,settings.
hrr_bins);

        context=[tempstruct(:,j).aspect_ratio1;
tempstruct(:,j).aspect_ratio2; tempstruct(:,j).aspect_ratio3;
tempstruct(:,j).bulbosity]';
        %user will define which aspect ratio to use
        %context(:,4) is bulbosity
        if and(settings.aspect_ratio ==0, settings.bulbosity ==0) %no context
            context = [];
        elseif and(settings.aspect_ratio == 0, settings.bulbosity >0) %No
aspect ratio
            context = context(:,4);
        elseif and(settings.aspect_ratio > 0, settings.bulbosity == 0) %No
bulbosity
            context = context(:,settings.aspect_ratio);
        else %use both context data
            context = [context(:,settings.aspect_ratio),context(:,4)];
        end

        %puts the features back into the individual structures
        for k = 1:size(tempstruct,1)
            tempstruct(k,j).hrrfeatures = hrrfeatures(k,:);
            if ~isempty(context) %if context is no empty
                tempstruct(k,j).features = [hrrfeatures(k,:),context(k,:)];
            else %if context is empty, no context info being used
                tempstruct(k,j).features = hrrfeatures(k,:);
            end
        end
    end %end loop for polarity
    train_exemplars = [train_exemplars; tempstruct];
end %end loop for target class



function [test_exemplars]=Build_Test
%================================
%Set up the testing data. Establish features for each exemplar.
%John Situ
%AFIT/ENS
%Last Updated: 04Jan12
%===============================

tStart = tic; %start timer
%establish folder paths
addpath([pwd, '\data']);
addpath([pwd, '\Functions']);
%load settings
load([pwd, '\settings.mat']);

%load testing data file
load([pwd, '\data\testdata.mat']);
%create list of structures in testing data file
list=whos('-file', [pwd, '\data\testdata']);
numstruct = length(list);

%Define the number features from hrr profiles and wedge sizes
num_hrrfeatures = settings.num_hrrfeatures;

%setup testing data structure
test_exemplars = struct([]);
for i = 1:numstruct %loop through all the class structures
    tempstruct = struct;
    eval(['tempstruct = ' list(i).name ';']) %assign current structure to
tempstruct
    for j = 1:settings.num_polar %loop for all polarities
        %combine all hrrprofiles for each class&polarity into one matrix
        hrrprofiles = [tempstruct(:,j).hrr]';

        %call get_hrr_features function
        %input (hrrprofiles, num of features, bin parameter maxormean,
hrrbins)
        hrrfeatures =
get_hrr_features(hrrprofiles,num_hrrfeatures,settings.hrr_maxormean,settings.
hrr_bins);
```

```
        context=[tempstruct(:,j).aspect_ratio1;
tempstruct(:,j).aspect_ratio2; tempstruct(:,j).aspect_ratio3;
tempstruct(:,j).bulbosity]';
        %user will define which aspect ratio to use
        %context(:,4) is bulbosity
        if and(settings.aspect_ratio ==0, settings.bulbosity ==0) %no context
            context = [];
        elseif and(settings.aspect_ratio == 0, settings.bulbosity >0) %No
aspect ratio
            context = context(:,4);
        elseif and(settings.aspect_ratio > 0, settings.bulbosity == 0) %No
bulbosity
            context = context(:,settings.aspect_ratio);
        else %use both context data
            context = [context(:,settings.aspect_ratio),context(:,4)];
        end

        %puts the features back into the individual structures
        for k = 1:size(tempstruct,1)
            tempstruct(k,j).hrrfeatures = hrrfeatures(k,:);
            if ~isempty(context) %if context is no empty
                tempstruct(k,j).features = [hrrfeatures(k,:),context(k,:)];
            else %if context is empty, no context info being used
                tempstruct(k,j).features = hrrfeatures(k,:);
            end
        end
    end %end loop for polarity
    test_exemplars = [test_exemplars; tempstruct];
end %end loop for target class

time = toc(tStart);
%Display Done messagge
disp(['Built Testing Set in ' num2str(time) ' secs']);

time = toc(tStart);
%Display Done messagge
disp(['Built Training Set in ' num2str(time) ' secs']);



function [results]= Wedge_Test(exemplars,set)
%================================
%Test exemplars against the inlibrary class templates
%John Situ
%AFIT/ENS
%last update 04Jan12
%===============================

tStart = tic; %start timer
%establish folder paths
addpath([pwd, '\data']);
addpath([pwd, '\Functions']);
%load settings
load([pwd, '\settings.mat']);
eqvar = settings.eqvar;
```

```
%load training template file
load([pwd, '\data\traintemplate.mat']);

%Define the number of polorizations
num_inlib_targets = settings.num_inlib_targets;
num_polar = settings.num_polar;

template = train_template;
%find the number of examplars in the current data set
num_examplars = size(exemplars,1);
for i = 1:num_examplars %loop index for every exemplar
    for j = 1:num_polar %loop for all polarities
        aspect_angle = exemplars(i,j).aspect;%exemplar aspect angle
        %find the wedge aspect angle begins to, also 1 to left and 1 to right
        center_wedge = ceil(aspect_angle/settings.wedge_degrees);
        wedge_nums = find_wedge_nums(center_wedge, settings.num_wedges);
        exemp = exemplars(i,j).features;%exemplar features

        mahal_dist = zeros(num_inlib_targets, 2);
        cos_sim = zeros(1,num_inlib_targets);
        norm_prob = zeros(1,num_inlib_targets);
        for k = 1:num_inlib_targets %index for all library targets
            temp_dist = zeros(3,2);
            temp_sim = zeros(1,3);
            for z = 1:3 %loop index for the 3 possible wedges
                mu = template(k,wedge_nums(z),j).wedge_mu;
                sigma = template(k,wedge_nums(z),j).wedge_sigma;
                %mahalanobis Distance
                temp_dist(z,1)= discriminant_function(exemp, mu,
sigma,eqvar);
                temp_dist(z,2)= wedge_nums(z);
                %Cosine Similiarity function
                temp_sim(z) = find_cos_similarity(exemp, mu);
            end
            %find the wedge with the lowest mahalanobis distance
            [mahal_dist(k,1),wedge_index] = min(temp_dist(:,1));
            %stores the winning wedge number from each class
            mahal_dist(k,2) = temp_dist(wedge_index,2);
            %find the wedge with the lowest similarity score
            cos_sim(k) = min(temp_sim);
            %Class Probabilities
            mu = template(k,mahal_dist(k,2),j).wedge_mu;
            sigma = template(k,mahal_dist(k,2),j).wedge_sigma;
            norm_prob(k)= find_multi_norm(exemp, mu, sigma,settings.eqvar);
        end %end loop for current libary target

        %Class Posterior Probabilty
        if settings.prior == 1 %equal prior prob for all classes
            post_prob = norm_prob ./ sum(norm_prob);
        else %unequal prior prob for the classes
            post_prob = (settings.prior_prob .* norm_prob) ./ ...
                sum(settings.prior_prob .* norm_prob);
        end

        if settings.forced_decision_opt == 1;
```

```
            %find the winning class using the min mahalanobis distance
            [~, class_index]=min(mahal_dist(:,1));
        else
            %find winning class using the max posterior probabilities
            [~, class_index]=max(post_prob);
        end

        %store winner class and wedge number in exemplar struct
        exemplars(i,j).mahal_dist = mahal_dist(:,1);
        exemplars(i,j).classify = class_index;
        exemplars(i,j).wedge = mahal_dist(class_index,2);
        exemplars(i,j).actual_wedge = center_wedge;
        exemplars(i,j).cos_sim = cos_sim;
        exemplars(i,j).norm_prob = norm_prob;
        exemplars(i,j).post_prob = post_prob;
    end
end
results = exemplars;
time = toc(tStart);
%Display Done messagge
if strcmp(set, 'train')
    disp(['Training Done in ' num2str(time) ' secs']);
elseif strcmp(set, 'test')
    disp(['Testing Done in ' num2str(time) ' secs']);
else
    disp(['Wedge Test Done in ' num2str(time) ' secs']);
end




function [op1, op2, op3, op4]= Wedge_Matrix_OOL(...
    train_exemplars,test_exemplars)
%=========================================
%This function use results from training to establish a Out of Library
%matrix of correctly identified training targets, organized by their wedge
%number and class. For Example if there are 24 wedges and 10 class, the OOL
%matrix is a 10x24 matrix. The elements are the highest mahalanobis distance
%of the correctly identified class at each that aspect angle wedge.

%John Situ
%AFIT/ENS
%last update 31Jan12
%=========================================



%This praticular OOL Matrix function uses mulitple threshold values and
%labels each exemplar the highest threshold it exceeds.
tStart = tic;

load([pwd, '\settings.mat']);
num_inlib_targets = settings.num_inlib_targets;
num_wedges = settings.num_wedges;
OOL_method = settings.OOL_method;
train_alpha = settings.OOL_train_alpha;
```

```
test_alpha = settings.OOL_test_alpha;
num_polar = settings.num_polar;

%Creat the OOL matrix with the highest values from each class and wedge
num_train_exemplars = size(train_exemplars,1);
%OOL_Matrix = struct;
OOL = struct('dist',[],'mu',[], 'sigma', [], 'count', []);
%create one extra library for dummy spot, prevents demension exceeding
OOL(num_inlib_targets+1,num_wedges,num_polar).dist = 0;

%wedge accuracy rate
wedge_count = 0;
exemplar_count = 0;

for i = 1:num_train_exemplars %Loop for every training exemplar
    for j = 1:num_polar %Loop for every polarity
        %if training exemplar truly belongs in outoflibrary, skip exemplar
        if ismember(train_exemplars(i,j).number,settings.outlib_targets)
            continue; %skip loop
        end
        class = train_exemplars(i,j).classify; %what wedge the system thinks
its in
        wedge = train_exemplars(i,j).wedge; %what wedge the system thinks its
in
        true_wedge = train_exemplars(i,j).actual_wedge; %what wedge its
actually in
        exemplar_count = exemplar_count+1;
        if  wedge == true_wedge
            wedge_count = wedge_count+1;

        end
        dist = min(train_exemplars(i,j).mahal_dist);
        %If exemplar was correctly identified
        if class == train_exemplars(i,j).order_number
            %if the exemplar's is classified correctly
            %the winning mahalanobis distance is added to the target and
            %wedge list of distances.
            spot = size(OOL(class,true_wedge,j).dist,2);
            OOL(class,true_wedge,j).dist(spot+1)= dist;
        end
    end %end loop for polarity
end %end loop for current training exemplar
wedge_acc = wedge_count/exemplar_count;
op4 = wedge_acc;

%gather statistic on every list of distances in the matrix
%orgranize by target number, wedge number and polarity
for class = 1:num_inlib_targets
    for wedge = 1:num_wedges
        for j = 1:num_polar
            OOL(class,wedge,j).mu = mean(OOL(class,wedge,j).dist);
            OOL(class,wedge,j).sigma = std(OOL(class,wedge,j).dist);
            OOL(class,wedge,j).count = size(OOL(class,wedge,j).dist,2);
        end
    end
end
```

```
%Using OOL_Matrix, find if train exemplars are OOL or in-library
%Will store decision in exemplar structure, field 'OOL'
%There are k thresholds for OOL, labels 1,...k is 'OOL'
%label 0 is 'NOOL', or not out-of-library.
num_train_exemplars = size(train_exemplars,1);

for i = 1:num_train_exemplars
    for j = 1:num_polar %Loop for every polarity
        class = train_exemplars(i,j).classify;%class exemplar classified by
distance methods
        wedge = train_exemplars(i,j).wedge; %what wedge the system thinks its
in
        dist = min(train_exemplars(i,j).mahal_dist);
        %if winning class mahalanobis is greater than threshold*alpha level
        %or alpha percentile threshold (method 1 and method 2)
        %there are k alphas values, highest resulting threshold reached will
        %be the label given for exemplar.
        %identify using same polarization OOL matrix
        num_train_thresholds = length(settings.OOL_train_alpha);

        thresholds = zeros(1,num_train_thresholds);
        for k = 1:num_train_thresholds
            if OOL_method ==1 %percentile Method
                thresholds(k)= prctile(OOL(class,wedge,j).dist,
train_alpha(k),2);
            elseif OOL_method ==2 %Mean + Sigma Method
                thresholds(k) = OOL(class,wedge,j).mu + ...
                    OOL(class,wedge,j).sigma* train_alpha(k);
            elseif OOL_method ==3 %Mean + MAD method;
                thresholds(k) = OOL(class,wedge,j).mu +...
                        mad(OOL(class,wedge,j).dist) * train_alpha(k);
            end
            if dist > thresholds(k) %determine the highest threshold exemplar
exceeds
                %exemplar is OOL
                train_exemplars(i,j).OOL = k;
            end
        end
        if dist <= thresholds(settings.OOL_threshold) %if exemplar is below
main thresholds
                %exemplar is NOOL
                train_exemplars(i,j).OOL = 0;
        end
    end %end loop for polarity
end

%Using OOL_Matrix, find if train exemplars are OOL or in-library
%Will store decision in exemplar structure, field 'OOL'
%There are k thresholds for OOL, labels 1,...k is 'OOL'
%label 0 is 'NOOL', or not out-of-library.
num_test_exemplars = size(test_exemplars,1);

for i = 1:num_test_exemplars
    for j = 1:num_polar %Loop for every polarity
```

```
        class = test_exemplars(i,j).classify;%class exemplar classified by
distance methods
        wedge = test_exemplars(i,j).wedge;%what wedge the system thinks its
in
        dist = min(test_exemplars(i,j).mahal_dist);
        %if winning class mahalanobis is greater than threshold*alpha level
        %or alpha percentile threshold (method 1 and method 2)
        %there are k alphas values, highest resulting threshold reached will
        %be the label given for exemplar.
        %identify using same polarization OOL matrix
        num_test_thresholds = length(settings.OOL_test_alpha);

        thresholds = zeros(1,num_test_thresholds);
        for k = 1:num_test_thresholds
            if OOL_method ==1 %percentile Method
                thresholds(k)= prctile(OOL(class,wedge,j).dist,
test_alpha(k),2);
            elseif OOL_method ==2 %Mean + Sigma Method
                thresholds(k) = OOL(class,wedge,j).mu + ...
                    OOL(class,wedge,j).sigma* test_alpha(k);
            elseif OOL_method ==3 %Mean + MAD method;
                thresholds(k) = OOL(class,wedge,j).mu +...
                        mad(OOL(class,wedge,j).dist) * test_alpha(k);
            end
            if dist > thresholds(k) %determine the highest threshold exemplar
exceeds
                %exemplar is OOL
                test_exemplars(i,j).OOL = k;
            end
        end
        if dist <= thresholds(settings.OOL_threshold) %if exemplar is below
main thresholds
                %exemplar is NOOL
                test_exemplars(i,j).OOL = 0;
        end
    end %end loop for polarity
end

%function output values
op1 = train_exemplars;
op2 = test_exemplars;
op3 = OOL;

time = toc(tStart);
%Display Done message
if settings.display == 1
    disp(['OOL Done in ' num2str(time) ' secs']);
end




function [train_output, test_output]=Turnbaugh_NDEC(...
    train_exemplars, test_exemplars, metric_type, alpha)
%=========================================
%Using Turnbaugh's NDEC method, make a decision on if train exemplars
```

```
%are nondeclarable (NDEC) or declarable (DEC)
%Will store decision in exemplar structure, field 'NDEC
%1 = NDEC, 2 = DEC
%must input evalution metric parameter:
%1 = evaluate NDEC using Mahalonobis Distance
%========================================

tStart = tic;
load([pwd, '\settings.mat']);

%Find NDEC decision for training exemplars
num_train_exemplars = size(train_exemplars,1);
num_polar = settings.num_polar;

for i = 1:num_train_exemplars
    for j = 1:num_polar
        %Use Mahalonobis Distance
        if metric_type == 1
            metric_values = train_exemplars(i,j).mahal_dist; %load metric
values
            sorted_values = sort(metric_values); %sort values ascending order
        %Use Cosine Similiarity metric
        elseif metric_type ==2
            metric_values = train_exemplars(i,j).cos_sim;
            sorted_values = sort(metric_values); %sort values ascending order
        %Posterior Probabilities
        elseif metric_type ==3
            %since higher probability is better, need to sort by
            %descending order
            metric_values = train_exemplars(i,j).post_prob;
            sorted_values = sort(metric_values, 'descend'); %sort values
descending order
        end
        best1_value = sorted_values(1); %find lowest/highest value
        best2_value = sorted_values(2); %find 2nd lowest/highest value
        worst_value = sorted_values(settings.num_inlib_targets); %find
highest val
        range = abs(best1_value- worst_value);
        difference = abs(best1_value - best2_value);

        %if the difference between the best two metric values is less than
        %the threshold value(range*user defined alpha), then a declaration
        %cannot be made. NDEC label
        %there are k threshold values, highest threshold reach will
        %be the label given for exemplar.
        num_NDEC_thresholds = length(settings.NDEC_alpha);
        for k =1:num_NDEC_thresholds
            if  difference < (range*alpha(k))
                %exemplar is NDEC
                train_exemplars(i,j).NDEC = k;
            end
        end
        if difference >= (range*alpha(settings.NDEC_threshold)) %greater than
highest threshold
            %exemplar is DEC
            train_exemplars(i,j).NDEC = 0;
        end
    end
end

%Find NDEC decision for testing exemplars
num_test_exemplars = size(test_exemplars,1);
for i = 1:num_test_exemplars
    for j = 1:num_polar
        %Use Mahalonobis Distance
        if metric_type == 1
            metric_values = test_exemplars(i,j).mahal_dist; %load metric
values
            %Use Cosine Similiarity metric
        elseif metric_type ==2
            metric_values = test_exemplars(i,j).cos_sim;
            %Posterior Probabilities
        elseif metric_type ==3
            %since higher probability is better, need to take neg prob to be
            %consistent with the other two metrics
            metric_values = -1* test_exemplars(i,j).post_prob;
        end
        sorted_values = sort(metric_values); %sort values ascending order
        min1_value = sorted_values(1); %find lowest value
        min2_value = sorted_values(2); %find 2nd lowest value
        max_value = sorted_values(settings.num_inlib_targets); %find highest
val
        range = max_value - min1_value;
        difference = min2_value - min1_value;

        %if the difference between the best two metric values is less than
        %the threshold value(range*user defined alpha), then a declaration
        %cannot be made. NDEC label
        %there are k threshold values, highest threshold reach will
        %be the label given for exemplar.
        num_NDEC_thresholds = length(settings.NDEC_alpha);
        for k =1:num_NDEC_thresholds
            if  difference < (range*alpha(k))
                %exemplar is NDEC
                test_exemplars(i,j).NDEC = k;
            end
        end
        if difference >= (range*alpha(settings.NDEC_threshold)) %greater than
highest threshold
            %exemplar is DEC
            test_exemplars(i,j).NDEC = 0;
        end
    end
end

%function output
train_output = train_exemplars;
test_output = test_exemplars;

time = toc(tStart);
%Display Done messagge
if settings.display ==1
```

```
    disp(['NDEC Done in ' num2str(time) ' secs']);
end


function [PNN] = PNN_Train3(train_exemplars)
%================================
%Organize training data, build and train PNN network.
%This organize the data into wedges and creates a PNN for every wedge.
%Must run Build_Train.m first.
%John Situ
%AFIT/ENS
%last update 14Feb12
%===============================
tStart = tic; %start timer
%establish folder paths
addpath([pwd, '\data']);
addpath([pwd, '\Functions']);

%load settings
load([pwd, '\data\settings.mat']);

%Define the number of polorizations
num_inlib_targets = settings.num_inlib_targets;
num_wedges = settings.num_wedges;
num_polar = settings.num_polar;

%Define the PNN Spread parameter
PNN_spread = settings.PNN_spread;

%establish a empty structure for the data by wedges and polarity
for i = 1:num_wedges
    for j = 1:num_polar
        PNN(i,j).inlib_input = [];
        PNN(i,j).inlib_class = [];
    end
end

exemplars = train_exemplars;
%find the number of examplars in the current data set
num_examplars = size(exemplars,1);
for i = 1:num_examplars %loop index for every exemplar
    for j = 1:num_polar %loop for all polarities
        class = exemplars(i,j).order_number;%the ordered class number
        %1-5 Enemy 6-10 Friendly, 11-15 OOL
        if class <= num_inlib_targets;
            aspect_angle = exemplars(i,j).aspect;
            wedge = ceil(aspect_angle/settings.wedge_degrees);
            features = exemplars(i,j).features;
            PNN(wedge, j).inlib_input = [PNN(wedge,j).inlib_input, 
features'];
            PNN(wedge, j).inlib_class = [PNN(wedge,j).inlib_class, class];
        end
    end
end
```

```
%Train PNN
for i = 1:num_wedges
    for j = 1:num_polar
        PNN(i,j).inlib_class_vec = ind2vec(PNN(i,j).inlib_class);
        P = PNN(i,j).inlib_input;
        T = PNN(i,j).inlib_class_vec;
        net = newpnn(P, T, PNN_spread);
        PNN(i,j).net = net;
    end
end

time = toc(tStart);
disp(['PNN Built and Trained in ' num2str(time) ' secs']);




function [exemplars,PNN] = PNN_Test3(exemplars,PNN,data_set)
%================================
%Organize exemplars for test on PNN network.
%Must run PNN_Train.m first.
%John Situ
%AFIT/ENS
%last update 16Feb12
%===============================

tStart = tic; %start timer
%establish folder paths
addpath([pwd, '\data']);
addpath([pwd, '\Functions']);

%load settings
load([pwd, '\data\settings.mat']);

%Define the number of polorizations and classes
num_targets = settings.num_targets;
num_inlib_targets = settings.num_inlib_targets;
num_wedges = settings.num_wedges;
num_polar = settings.num_polar;

%establish a empty structure for the data polarity
PNN(1,1).actual_classes = [];
PNN(1,2).actual_classes = [];
PNN(1,1).predicted_classes = [];
PNN(1,2).predicted_classes = [];

%find the number of examplars in the current data set
num_examplars = size(exemplars,1);

for i = 1:num_examplars %loop index for every exemplar
    for j = 1:num_polar %loop for all polarities
        class = exemplars(i,j).order_number;%the ordered class number
        %1-5 Enemy 6-10 Friendly, 11-15 OOL
        %create array of actual classes for confusion matrix
        PNN(1,j).actual_classes = [PNN(1,j).actual_classes, class];
        aspect_angle = exemplars(i,j).aspect;
        %find the wedge aspect angle begins to, also 1 to left and 1 to right
        center_wedge = ceil(aspect_angle/settings.wedge_degrees);
        wedge_nums = find_wedge_nums(center_wedge, num_wedges);
        features = exemplars(i,j).features;
        PNN_results = zeros(3,2);
        for k = 1:3
            %Find the predicted class vector
            prediction = sim(PNN(wedge_nums(k),j).net, features');
            PNN_results(k,1) = vec2ind(prediction);
            %Find max score value from predicted class or datapoint
            PNN_results(k,2)= max(radbas(netprod(dist(...
                PNN(wedge_nums(k),j).net.IW{1,1},...
                exemplars(i,j).features'),PNN(wedge_nums(k),j).net.b{1})));
        end
        %Find the mode class, the most frequent;
        [mode_class, freq] = mode(PNN_results(:,1));
        %Fine the closest class, the max score value
        [~, max_index] = max(PNN_results(:,2));
        max_class = PNN_results(max_index,1);

        exemplars(i,j).PNN_wedges = wedge_nums;
        exemplars(i,j).PNN_classes = PNN_results(:,1)';
        exemplars(i,j).PNN_dists = PNN_results(:,2)';

        if freq == 1
            %if each wedge gives different classes, use max score
            exemplars(i,j).PNN_decision = max_class;
        elseif freq ==3
            %if all 3 wedges agree, use mode
            exemplars(i,j).PNN_decision = mode_class;
        elseif freq ==2
            %if 2 of 3 agree
            exemplars(i,j).PNN_decision = mode_class;
        end

        %create array of predicted classes for confusion matrix
        PNN(1,j).predicted_classes = [PNN(1,j).predicted_classes,...
            exemplars(i,j).PNN_decision];
    end
end

%build confusion matrix
for j = 1:num_polar
    if strcmp(data_set, 'train')
        [PNN(1,j).train_CM, ~] = confusionmat(PNN(1,j).actual_classes,...
            PNN(1,j).predicted_classes);
        PNN(1,j).train_CM(:,num_inlib_targets+1:num_targets)=[];
    elseif strcmp(data_set, 'test')
        [PNN(1,j).test_CM, ~] = confusionmat(PNN(1,j).actual_classes,...
            PNN(1,j).predicted_classes);
        PNN(1,j).test_CM(:,num_inlib_targets+1:num_targets)=[];
    end
end
```

```
time = toc(tStart);
%Display Done messagge
if strcmp(data_set, 'train')
    disp(['PNN Training set Done in ' num2str(time) ' secs']);
elseif strcmp(data_set, 'test')
    disp(['PNN Testing set Done in ' num2str(time) ' secs']);
else
    disp(['PNN Test Done in ' num2str(time) ' secs']);
end




function [CPD]=get_BNT_CPD1(train_exemplars)
%============================================
%Function builds the BNT CPD for used in final_BNT1.m
%============================================
load([pwd, '\data\settings.mat']);
num_inlib_targets = settings.num_inlib_targets;
OOL_target_num = num_inlib_targets+1;
num_OOL_threshold = length(settings.OOL_test_alpha);
num_NDEC_threshold = length(settings.NDEC_alpha);
num_polar = settings.num_polar;

class_sums = zeros(num_inlib_targets+1,num_polar);
num_train_exemplars = size(train_exemplars,1);

AR_bins = settings.AR_bins;
num_AR_bins = settings.num_AR_bins;
bulb_bins = settings.bulb_bins;
num_bulb_bins = settings.num_bulb_bins;

OOLtotal = zeros(num_inlib_targets+1,num_OOL_threshold+1,num_polar);
NDECtotal = zeros(num_inlib_targets+1,num_NDEC_threshold+1,num_polar);
Forcedtotal = zeros(num_inlib_targets+1,num_inlib_targets,num_polar);
ARtotal = zeros(num_inlib_targets+1,num_AR_bins,num_polar);
Bulbtotal = zeros(num_inlib_targets+1,num_bulb_bins,num_polar);
PNNtotal = zeros(num_inlib_targets+1,num_inlib_targets,num_polar);

%Etotal = zeros(num_inlib_targets+1,num_aspect_ratio_bins);
for i = 1:num_train_exemplars
    for j = 1: num_polar % loop for each polarity
        %find the ordered class number of exemplar
        %1:5 Enemy, 6:10 Friend, 11:15 OOL
        class = train_exemplars(i,j).order_number;
        %non-library targets will be group into one class "OOL target"
        %if there are 10 in-lib targets, then OOL target class is 11
        if ~ismember(class, 1:num_inlib_targets)
            class = OOL_target_num;
        end

        %find the decision of the exemplar
        label = train_exemplars(i,j).classify; %forced decision
        OOL = train_exemplars(i,j).OOL; %OOL decision
        NDEC = train_exemplars(i,j).NDEC; %NDEC decision
```

```
PNN_label = train_exemplars(i,j).PNN_decision; %PNN decision
%find the context info of the exemplar
if settings.aspect_ratio ==1
    AR = train_exemplars(i,j).aspect_ratio1;
elseif settings.aspect_ratio ==2
    AR = train_exemplars(i,j).aspect_ratio2;
elseif settings.aspect_ratio ==3
    AR = train_exemplars(i,j).aspect_ratio3;
else
    AR = 1; %Arbitary number
    %Will result in CPD to be zero matrix
end

if settings.bulbosity == 1
    bulb = train_exemplars(i,j).bulbosity;
else
    bulb = 1;%Arbitary number
    %Will result in CPD to be zero matrix
end

%count the number of each class
class_sums(class,j) = class_sums(class,j) +1;

if OOL > 0 %OOL
    OOLtotal(class,OOL,j) = OOLtotal(class,OOL,j) +1;
elseif OOL == 0 %NOOL
    OOLtotal(class,num_OOL_threshold+1,j) = ...
        OOLtotal(class,num_OOL_threshold+1,j) +1;
end

if NDEC > 0 %NDEC
    NDECtotal(class,NDEC,j) = NDECtotal(class,NDEC,j) +1;
elseif NDEC == 0 %DEC
    NDECtotal(class,num_NDEC_threshold+1,j) = ...
        NDECtotal(class,num_NDEC_threshold+1,j) +1;
end

%Forced Decision matrix
Forcedtotal(class,label,j) = Forcedtotal(class,label,j)+1;

%PNN Decision matrix
PNNtotal(class,PNN_label,j) = PNNtotal(class,PNN_label,j)+1;

%ONLY add Contextual CPD data for inlibray classes.
%Skip for OOL classes.
if class <= num_inlib_targets
    %aspect ratio matrix
    for k = 1:num_AR_bins
        if AR <= AR_bins(k)
            %find the bin AR value belongs to.
            %increament matrix and %termainate loop.
            ARtotal(class,k,j) = ARtotal(class,k,j)+1; break; %
        end
    end
    %bulbosity matrix
```

```
            for k = 1:num_bulb_bins
                if bulb <= bulb_bins(k)
                    %find the bin bulbosity value belongs to,
                    %increament matrix and %termainate loop.
                    Bulbtotal(class,k,j) = Bulbtotal(class,k,j)+1; break;
                end
            end
        end
    end %end loop for polarity
end %end loop for exemplar

Truthprob = class_sums/sum(class_sums);
OOLprob = zeros(num_inlib_targets+1,num_OOL_threshold+1,num_polar);
NDECprob = zeros(num_inlib_targets+1,num_NDEC_threshold+1,num_polar);
Forcedprob = zeros(num_inlib_targets+1,num_inlib_targets,num_polar);
ARprob = zeros(num_inlib_targets+1,num_AR_bins,num_polar);
Bulbprob = zeros(num_inlib_targets+1,num_bulb_bins,num_polar);
PNNprob = zeros(num_inlib_targets+1,num_inlib_targets,num_polar);

for i = 1:(num_inlib_targets + 1)
    for j = 1: num_polar
        OOLprob(i,:,j) = OOLtotal(i,:,j)/class_sums(i,j);
        NDECprob(i,:,j) = NDECtotal(i,:,j)/class_sums(i,j);
        Forcedprob(i,:,j) = Forcedtotal(i,:,j)/class_sums(i,j);
        ARprob(i,:,j) = ARtotal(i,:,j)/class_sums(i,j);
        Bulbprob(i,:,j) = Bulbtotal(i,:,j)/class_sums(i,j);
        PNNprob(i,:,j) = PNNtotal(i,:,j)/class_sums(i,j);
    end
end

%Contextual CPD for OOL classes will be an average of
%in library class CPD row values
ARprob(num_inlib_targets+1, :, 1) = ...
    sum(ARtotal(1:num_inlib_targets,:,1),1)/...
    sum(class_sums(1:num_inlib_targets,1));
ARprob(num_inlib_targets+1, :, 2) = ...
    sum(ARtotal(1:num_inlib_targets,:,2),1)/...
    sum(class_sums(1:num_inlib_targets,2),1);
Bulbprob(num_inlib_targets+1, :, 1) =...
    sum(Bulbtotal(1:num_inlib_targets,:,1),1)/...
    sum(class_sums(1:num_inlib_targets,1),1);
Bulbprob(num_inlib_targets+1, :, 2) =...
    sum(Bulbtotal(1:num_inlib_targets,:,2),1)/...
    sum(class_sums(1:num_inlib_targets,2),1);

%output CPD
CPD = {Truthprob, OOLprob(:,:,1), OOLprob(:,:,2), ...
    NDECprob(:,:,1),NDECprob(:,:,2),Forcedprob(:,:,1),Forcedprob(:,:,2),...
    ARprob(:,:,1),ARprob(:,:,2),Bulbprob(:,:,1),Bulbprob(:,:,2),...
    PNNprob(:,:,1),PNNprob(:,:,2)};




function [output] = final_BNT1(test_exemplars, CPD)
```

```
%================================
%function uses training results and construct a Bayesian Network
%Uses BNT function
%Creates posterior probabilities of classes

%the network will be same for all conditions, if there's no information, it
%will be assigned uniform number. It will not have any affect on the outcome
%This is to aviod having multiple BNT functions for different networks.
%If there's only one polarity, the second polarity nodes will be
%depulicates of the first polarity. If there's contextual information not
%used in the framework, the invalid nodes will be assigned 1s.

%This is the full 13 node BNT, it has 12 dependent nodes:
%OOL, NDEC, Forced Decisions, PNN Decision, AR and Bulbosity Info nodes

%John Situ
%Last updated:18Feb11
%===============================

tStart = tic; %start timer

%BEFORE EXECUTION, MAKE SHOULD THAT "BNT Code" FOLDER IS ADDED TO MATLAB
PATH!!!
addpath(genpath([pwd, '\BNT Functions']));
load([pwd, '\data\settings.mat']);

%load the parameters from settings.m
num_polar = settings.num_polar;
num_inlib_targets = settings.num_inlib_targets;
num_OOL_threshold = length(settings.OOL_test_alpha);
num_NDEC_threshold = length(settings.NDEC_alpha);
AR_bins = settings.AR_bins;
num_AR_bins = settings.num_AR_bins;
bulb_bins = settings.bulb_bins;
num_bulb_bins = settings.num_bulb_bins;

%define BNT network and nodes.
N = 13;
dag = zeros(N,N);
Truth = 1; HOOL = 2; VOOL = 3; HNDEC = 4; VNDEC = 5;
HForced = 6; VForced = 7; HAR = 8; VAR =9; HBulb =10; VBulb= 11;
HPNN = 12; VPNN = 13;

dag(Truth,HOOL) = 1;
dag(Truth,VOOL) = 1;
dag(Truth,HNDEC) = 1;
dag(Truth,VNDEC) = 1;
dag(Truth,HForced) = 1;
dag(Truth,VForced) = 1;
dag(Truth,HAR) = 1;
dag(Truth,VAR) = 1;
dag(Truth,HBulb) = 1;
dag(Truth,VBulb) = 1;
dag(Truth,HPNN) = 1;
dag(Truth,VPNN) = 1;
```

```
discrete_nodes = 1:N;
node_sizes = [num_inlib_targets+1,...
    num_OOL_threshold+1, num_OOL_threshold+1,...
    num_NDEC_threshold+1, num_NDEC_threshold+1,...
    num_inlib_targets, num_inlib_targets,...
    num_AR_bins, num_AR_bins, num_bulb_bins, num_bulb_bins,...
    num_inlib_targets, num_inlib_targets];

bnet = mk_bnet(dag, node_sizes, discrete_nodes);
bnet.CPD{Truth} = tabular_CPD(bnet, Truth, reshape(CPD{1},1,[]));
bnet.CPD{HOOL} = tabular_CPD(bnet, HOOL, reshape(CPD{2},1,[]));
bnet.CPD{VOOL} = tabular_CPD(bnet, VOOL, reshape(CPD{3},1,[]));
bnet.CPD{HNDEC} = tabular_CPD(bnet, HNDEC, reshape(CPD{4},1,[]));
bnet.CPD{VNDEC} = tabular_CPD(bnet, VNDEC, reshape(CPD{5},1,[]));
bnet.CPD{HForced} = tabular_CPD(bnet, HForced, reshape(CPD{6},1,[]));
bnet.CPD{VForced} = tabular_CPD(bnet, VForced, reshape(CPD{7},1,[]));
bnet.CPD{HAR} = tabular_CPD(bnet, HAR, reshape(CPD{8},1,[]));
bnet.CPD{VAR} = tabular_CPD(bnet, VAR, reshape(CPD{9},1,[]));
bnet.CPD{HBulb} = tabular_CPD(bnet, HBulb, reshape(CPD{10},1,[]));
bnet.CPD{VBulb} = tabular_CPD(bnet, VBulb, reshape(CPD{11},1,[]));
bnet.CPD{HPNN} = tabular_CPD(bnet, HPNN, reshape(CPD{12},1,[]));
bnet.CPD{VPNN} = tabular_CPD(bnet, VPNN, reshape(CPD{13},1,[]));

evidence = cell(1,N);

engine = jtree_inf_engine(bnet);

forced_dec = zeros(1,num_polar);
OOL = zeros(1,num_polar);
NDEC = zeros(1,num_polar);
AR = zeros(1,num_polar);
bulb = zeros(1,num_polar);
PNN = zeros(1,num_polar);

num_test_exemplars = size(test_exemplars,1);
for i = 1 :num_test_exemplars
    for j = 1:num_polar % loop for each polarity
        forced_dec(j) = test_exemplars(i,j).classify; %forced decision
        OOL(j) = test_exemplars(i,j).OOL; %OOL decision
        NDEC(j)= test_exemplars(i,j).NDEC; %NDEC decision
        PNN(j)= test_exemplars(i,j).PNN_decision;

        %%%%%
        %k is the number of threshold user defined for each classifer
        %node events 1-k is for OOL, event k +1 is NOOL
        if OOL(j) == 0 %exemplar label is NOOL, need id event as k+1
            OOL(j) = num_OOL_threshold+1;
        end

        %node events 1-k is for NDEC, event k +1 is DEC
        if NDEC(j) == 0 %exemplar label is DEC, need id event as k+1
            NDEC(j) = num_NDEC_threshold+1;
        end
        if settings.aspect_ratio ==1
            AR(j) = test_exemplars(i,j).aspect_ratio1;
        elseif settings.aspect_ratio ==2
            AR(j) = test_exemplars(i,j).aspect_ratio2;
        elseif settings.aspect_ratio ==3
            AR(j) = test_exemplars(i,j).aspect_ratio3;
        else
            AR(j) = 0;
        end

        if settings.bulbosity == 1
            bulb(j) = test_exemplars(i,j).bulbosity;
        else
            bulb(j) = 0;
        end

        %find the bin aspect ratios belong to
        for k = 1:num_AR_bins
            if AR(j) <= AR_bins(k)
                %find the bin AR value belongs to.
                %assign bin number and termainate loop.
                AR(j) = k; break; %
            end
        end
        %find the bin bulbosity belong to
        for k = 1:num_bulb_bins
            if bulb(j) <= bulb_bins(k)
                %find the bin bulbosity value belongs to,
                %assign bin number and termainate loop.
                bulb(j) = k; break;
            end
        end
    end%end loop for polarity

    %given BNT the exemplar's evidence
    evidence{HOOL} = OOL(1);
    evidence{HNDEC} = NDEC(1);
    evidence{HForced} = forced_dec(1);
    evidence{HPNN} = PNN(1);
    if AR(1) > 0; evidence{HAR} = AR(1); end
    if bulb(1) >0 ; evidence{HBulb} = bulb(1); end

    if num_polar ==1 %if there's only polarity, just double the evidence
        evidence{VOOL} = OOL(1);
        evidence{VNDEC} = NDEC(1);
        evidence{VForced} = forced_dec(1);
        evidence{VPNN} = PNN(1);
        if AR(1) >0; evidence{VAR} = AR(1); end
        if bulb(1) >0 ;evidence{VBulb} = bulb(1); end
    end
    if num_polar ==2 %if there's 2 polarity, given the second set of evidence
        evidence{VOOL} = OOL(2);
        evidence{VNDEC} = NDEC(2);
        evidence{VForced} = forced_dec(2);
        evidence{VPNN} = PNN(2);
        if AR(2) > 0;evidence{VAR} = AR(2); end
        if bulb(2) >0 ;evidence{VBulb} = bulb(2); end
    end

    [engine, ~] = enter_evidence(engine, evidence);
    marg = marginal_nodes(engine, Truth);
    p = marg.T;
    [~, BNT_decision] = max(p);
    test_exemplars(i,1).BNT_prob = p;
    test_exemplars(i,2).BNT_prob = p;
    test_exemplars(i,1).BNT_decision = BNT_decision;
    test_exemplars(i,2).BNT_decision = BNT_decision;
end

output = test_exemplars;

time = toc(tStart);
%Display Done messagge
disp(['BNT Done in ' num2str(time) ' secs']);
```

Bibliography

[1] Albrecht, Timothy W. *Combat Identification With Sequential Observations, Rejection Option, and Out-Of-Library Target.* Ph.D. thesis, AFIT, Air Force Institute of Technology (AETC), Wright-Patterson AFB OH, 2005.

[2] Bo, Li and Gao Xiaoguang. "Bayesian Networks for Intelligent Decision of Airborne Weapon System". *Innovative Computing, Information and Control, 2006. ICICIC '06. First International Conference on*, volume 3, 390 –393. 30 2006-sept. 1 2006.

[3] C.K.Chow. "On Optimum Recognition Error and Reject Tradeoff", *IEEE Transactions on Information Theory*, 16(1):41–46, January 1970.

[4] Corr, D.G., A. Walker, U. Benz, I. Lingenfelder, and A. Rodrigues. "Classification of urban SAR imagery using object oriented techniques". *Geoscience and Remote Sensing Symposium, 2003. IGARSS '03. Proceedings. 2003 IEEE International*, volume 1, 188 – 190 vol.1. july 2003.

[5] Dasarathy, B.V. "Sensor fusion potential exploitation-innovative architectures and illustrative applications", *Proceedings of the IEEE*, 85(1):24 –38, jan 1997. ISSN 0018-9219.

[6] of Defense (Science, Deputy Under Secretary and Technology). *Joint Warfighting Science and Technology Plan.* Technical report, Department of Defense, February 2000.

[7] Dilger, Robert and Pierre Sprey. "Reversing the Decay of American Air Power", March 2009. URL `http://www.dodbuzz.com/2009/03/02/insurgents-offer-tough-air-critique/`.

[8] Duda, Richard O., Peter E. Hart, and David G. Stork. *Pattern Classification.* Wiley-Interscience; 2 edition, October 2000.

[9] Dunnigan, James F. *How to Make War (Fourth Edition): A Comprehensive Guide to Modern Warfare in the Twenty-first Century.* Harper Perennial, April 2003.

[10] Fishel, Judy. "Length of Waves on the Electromagnetic Spectrum", Aug 2011. URL `http://www.breakthroughlearningcollege.com/visual-processing/diagrams/diagrams-2/`.

[11] Force, Headquarters United States Air. *The Air War Over Serbia: Aerospace Power in Operation Allied Force.* Technical report, United States Air Force, April 2000.

[12] Franceschetti, Giorgio and Riccardo Lanari. *Synthetic aperture radar processing*. CRC Press, Boca Raton, Florida, 1999.

[13] Friend, Mark A. *Combat Identification with Synthetic Aperture Radar, Hidden Markov Models and Information Theory*. Ph.D. thesis, AFIT, Air Force Institute of Technology (AETC), Wright-Patterson AFB OH, 2007.

[14] Fumera, Giorgio, Fabio Roli, and Giorgio Giacinto. "Reject Option with Multiple Thresholds", *Pattern Recognition*, 33:2099–2101, 2000.

[15] Hall, David L. and James Llinasl. "An Introduction to Multisensor Data Fusion", *Proceedings of the IEEE*, 85(1):6–23, January 1997.

[16] Ho, Tin Kam and Jonathan J. Hull. "Decision Combination in Multiple Classifier Systems", *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 16:66–75, January 1994.

[17] Jain, A.K., R.P.W. Duin, and Jianchang Mao. "Statistical Pattern Recognition: A Review", *IEEE Transactions on Pattern Analysis and Machine Inteligence*, 22:4–37, 2000.

[18] Kuiying, Yin, Liu Hongwei, Hu Liping, and Jin Lin. "Marker-controlled SAR image segmentation algorithm". *Radar Conference, 2009 IET International*, 1 –5. april 2009.

[19] Laine, Trevor I. *Optimization of Automatic Target Recognition with a Reject Option Using Fusion and Correlated Sensor Data*. Ph.D. thesis, AFIT, Air Force Institute of Technology (AETC), Wright-Patterson AFB OH, 2005.

[20] Landgrebe, David A. "Hyperspectral Image Data Analysis", *IEEE Signal Processing Magazine*, 17–28, January 2002.

[21] Leap, Nathan J. *A Confidence Paradigm For Classification Systems*. Ph.D. thesis, AFIT, Air Force Institute of Technology (AETC), Wright-Patterson AFB OH, 2008.

[22] MATLAB. *version 7.10.0 (R2010a)*. The MathWorks Inc., Natick, Massachusetts, 2010.

[23] Messer, Adam J. *Contextual Detection of Anomalies within Hyperspectral Images*. Master's thesis, AFIT, Air Force Institute of Technology (AETC), Wright-Patterson AFB OH, march 2011.

[24] Murphy, Kevin. "Bayes Net Toolbox for Matlab", October 2007. URL `http://code.google.com/p/bnt/`.

[25] Nicoli, Louis P. and Georgios C. Anagnostopoulos. “Shape-based Recognition of Targets in Synthetic Aperture Radar Images using Elliptical Fourier Descriptors”, *Automatic Target Recognition XVIII*, 6967, April 2008.

[26] Ruta, Dymitr and Bogdan Gabrys. “An Overview of Classifier Fusion Methods”, *Computing and Information Systems*, 7:1–10, 2000.

[27] Samuel W. McCandless, Jr. and Christopher R. Jackson. “Chapter 17: Principles of Synthetic Aperture Radar”. *Synthetic Aperture Radar: Marine User’s Manual*. US Department of Commerce, National Oceanic and Atmospheric Adminstration, 2004.

[28] Smetek, Timothy E. and Kenneth W. Bauer Jr. “A Comparison of Mulivariate Outlier Detection Methods For Finding Hyperspectral Anomoalies”, *Military Operations Research*, 13(4):19–39, 2008.

[29] Sullivan, Roger. “Synthetic Aperture Radar”. *Radar Handbook*. McGraw-Hill, 2008.

[30] Toussaint, Godfried T. “The use of Context in Pattern Recogition”, *Pattern Recognition*, 10:189–204, 1978.

[31] Turnbaugh, Michael A. *A Hybrid Temple-Based Composite Classification System*. Ph.D. thesis, AFIT, Air Force Institute of Technology (AETC), Wright-Patterson AFB OH, 2008.

[32] Williams, Robert and John Westerkamp. “Automatic Target Recognition of Time Critical Moving Targets using 1D High Range Resolution (HRR) Radar”, *IEEE AES Systems Magazine*, 37–43, April 2000.

[33] Williams, Robert and John J. Westerkamp. “Analysis of a 1-D HRR moving target ATR”, *SPIE*, 3721(0277):413–424, 1999.

[34] Yang, He, Ben Ma, Qian Du, and Liangpei Zhang. “Comparison of spectral-spatial classification for urban hyperspectral imagery with high resolution”. *Urban Remote Sensing Event, 2009 Joint*, 1 –5. may 2009.

[35] Zaart, Ali El, Djemel Ziou, Shengrui Wang, and Qingshan Jiang. “Segmentation of SAR images”, *Pattern Recognition Society*, 35:713–724, 2002.

[36] Zhang, Liangpei, Xin Huang, Bo Huang, and Pingxiang Li. “A Pixel Shape Index Coupled with Spectral Information for Classification of High Spatial Resolution Remotely Sensed Imagery”, *IEEE Transaction on Geoscience and Remote Sensing*, 44(10):2950–2961, Oct 2006.

Vita

Captain John X. Situ is from Sugar Land, TX. In May 2006, he graduated from the University of Texas at Austin with a Bachelor of Science degree in Applied Mathematics. John was commissioned into the US Air Force through AFROTC Detachment 825.

Captain Situ's first assignment was at the Air Force Operational Test and Evaluation Center(AFOTEC) Detachment 5 at Edwards AFB, California. John was the Lead Analyst for Operational Test programs such as C-130J 56M Radar Warning System, C-27 Defense Weapon Systems Testing and C-5 Reliability Enhancement and Re-engining Program (RERP).

After graduation from AFIT, Capt Situ's next assignment will be at the Air Force Personnel Center at Randolph AFB, TX

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 074-0188*

1. REPORT DATE *(DD-MM-YYYY)* 03-22-2012	**2. REPORT TYPE** **Master's Thesis**	**3. DATES COVERED** *(From – To)* Aug 2010 - Mar 2012

4. TITLE AND SUBTITLE Combat Identification of Synthetic Aperture Radar Images using Contextual Features and Bayesian Belief Networks	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Situ, John, X. Capt, USAF	**5d. PROJECT NUMBER**
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Street, Building 642 WPAFB OH 45433-7765	**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT-OR-MS-ENS-12-24
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) “Intentionally left blank”	**10. SPONSOR/MONITOR’S ACRONYM(S)**
	11. SPONSOR/MONITOR’S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES
“Intentionally left blank”

14. ABSTRACT

Given the nearly infinite combination of modifications and configurations for weapon systems, no two targets are ever exactly same. SAR imagery and associated High Range Resolution (HRR) profiles of even the same target will both have different signatures when viewed from different angles. To overcome this challenge, data from a wide range of aspect and depression angles must be used to train pattern recognition algorithms. Alternatively, features invariant to aspect and depression angle must be found. This research uses simple segmentation algorithms and multivariate analysis methods to extract contextual features from SAR imagery. These features used in conjunction with HRR features improve classification accuracy at similar or extended operating conditions. Classification accuracy improvements achieved through Bayesian Belief Networks and the direct use of the contextual features in a template matching algorithm are demonstrated using a General Dynamics Data Collection System SAR data set.

15. SUBJECT TERMS

Combat Identification(CID), Synthetic Aperture Radar(SAR), Contexual Features, Bayesian Belief Networks, Decision-Level Fusion, Automated Target Recognition(ATR)

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Dr. Mark A. Friend, Lt Col, USAF
a. REPORT U	**b. ABSTRACT** U	**c. THIS PAGE** U	UU	133	**19b. TELEPHONE NUMBER** *(Include area code)* (937)255-3636, ext 4624; e-mail: Mark.Friend@afit.edu